EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

Software Development Division

ESO/ALMA Software

NG/AMS

Next Generation Archive Management System

User's Manual

Doc. No.: VLT-MAN-ESO-19400-2739

Issue: 4

Date: 28/12/2009

	Name	Date	Signature
Prepared:	J.Knudstrup	28/12/2009	•
Approved:	A.Wicenec, T.Bierwirth	/ /2010	•
Released:	M.Peron	/ /2010	•

CHANGE RECORD

ISSUE	DATE	SECTION/PAGE AFFECTED	REASON/INITIATION DOCUMENTS/REMARKS
1/Prep 1	29/01/2002	All	First issue.
1/Prep 2	05/03/2002	All	Added comment after internal review.
2/Prep 1	08/01/2003	All	Major update with new features.
3/Prep 1	06/08/2004	All	Update with new features + improved explanations and more info/guidelines.
4/Prep1	28/12/2009	All	Updated with new features: - User Command Plug-in. - User Service Plug-in. - Cache Archive Service. - Mirroring Service. In addition a general overhaul was carried out.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. About this Guide

1.1 Purpose & Scope

This document is the user's manual for the Next Generation Archive Management System (NG/AMS). NG/AMS is the SW for the Next Generation Archive System [2]. It is in charge of the handling of storage media and of archiving and retrieving data files to/from an NGAS Archive. Numerous other services are provided for carrying out the daily operation of an NGAS Archive System.

This manual contains the information needed for configuring and operating NG/AMS. It is also described how to enhance the system with new features by adding various types of plug-ins. These plug-ins are small Python functions with a specific interface and a specific set of tasks.

The audience of this document is NGAS Operators who perform archiving and retrieval of data files into/from NGAS and other maintenance actions needed to maintain the data in a good condition. However, also more advanced users who need to tune and adapt the system by changing the configuration will find the necessary information in this document. Finally support for the very advanced user is provided. The latter type of user is the user who adds or changes functionality of the system by providing new plug-ins or changing existing ones.

1.2 How to Read this Manual

For the user unknown to NG/AMS it is recommended to read this chapter and chapter 2 to get an overview of the manual and of NG/AMS and its features. For more specific issues it is suggested to check the index or the table of contents and read the referenced sections in connection with these issues.

It is recommended to have a WEB browser available when reading the manual, since many references to links are used in the explanations.

The following conventions are used in this manual:

Item	*Description*
<…>	A name in brackets indicates a substitution of the brackets + the name with the contents of the object referred by the name.
<text>	Courier font for examples of source code files and configuration files. In addition this font is used for commands as they must be types on the shell.
"<name>"	Names of SW modules, classes, methods, functions, files etc., are contained in quotes.
<element>[.<element>]	Reference to an XML element.
<element>[.<element>]:<attribute>	Used to refer to a specific attribute in an XML document, e.g.: "NgamsCfg.Server:PortNo".
CFG: <configuration component>	Reference to an element/attribute in the NG/AMS Configuration. For detailed information about the NG/AMS Configuration, consult chapter 6.
DB: <DB column>	Refers to a DB column. The reference may also be given as: "[<db>.][<table>.]<column".
FILE: <filename>	Reference to a file (within the NG/AMS SW Package).
FUNCTION:/METHOD: <name>	Reference to a specified function or method provided within the NG/AMS Package.

Table 1: Conventions used in this manual.

Some sections are dedicated to the more advanced users of NG/AMS. These sections are marked with "EXPERT". A 'normal user' may want to skip these sections.

Important information is presented in info text boxes like this:

	*<Information>*

Tips concerning the operation are presented in text boxes like this:

	*<Information>*

The last chapter (16) contains a quick reference to the commands supported by NG/AMS.

Throughout the document, references to the online documentation of NG/AMS are contained in the text as follows:

http://<Host>:<Port>/ngams.ngamsCClient.ngamsCClientLib.html

whereby it is assumed that an instance of the "pydoc" utility is running in 'daemon mode' (e.g. "pydoc –p 7575"):

http://jewel2.hq.eso.org:7575/ngams.ngamsCClient.ngamsCClientLib.html

1.3 How to Get Help and/or Report Problems with NG/AMS or this Manual

In case problems are encountered using NG/AMS, bug/problem reports can be submitted via email to NGAS Support Team:

ngast@eso.org

This also goes for questions and other assistance needed in connection with the usage and enhancement of the system

1.4 Disclaimer

Although great efforts have been invested in designing robust interfaces for the NG/AMS SW e.g. when it comes to the HTTP communication protocol, various XML document formats, and the interfaces of the APIs provided, it should be mentioned that NG/AMS is an advanced SW system operating in a complex environment and minor changes may have to be introduced in the various interfaces from time to time. It will however be attempted to limit the amount of such changes to an absolute minimum to guarantee a high level of backwards compatibility. See also chapter **Error! Reference source not found.**, NG/AMS License Conditions.

In general it is attempted to keep NG/AMS compatible with previous versions. However, in cases where it is necessary to change interfaces etc. to comply with standards, it may be that minor backwards incompatibilities are introduced.

1.5 Reference Documents

The following documents contain additional information and are referenced in the text:

Reference	Document Number	Issue	Date	Title
[1]	VLT-SPE-ESO-19400-2534	1	22/06/2001	"DFS Software, Next Generation/Archive Management System", Design Description, J.Knudstrup.
[2]	VLT-PLA-ESO-19400-3100	1	22/07/2003	"DFS Software, NGAS Acceptance Test Plan & Hands-On Tutorial", Test Plan, J.Knudstrup.
[3]	VLT-MAN-19400-3103	1	28/07/2003	"DFS Software, NGAS Operations & Troubleshooting Guide", User Manual, J.Knudstrup.

Table 2: Reference documents.

1.6 Acronyms

The following abbreviations and acronyms are used in this document:

DB	Database
DAPI	Data Archiving Plug-In
DPPI	Data Processing Plug-In
DTD	Document Type Definition
HDD	Hard Disk Drive
HW	Hardware
N/A	Not Applicable

NGAS	Next Generation Archive System
NGAS DB	NGAS (Data Holding) DB
NG/AMS	NGAS Archive Management System
OS	Operating System
SW	Software
XML	Extensible Markup Language

Table 3: Acronyms used in the NG/AMS User's Manual.

1.7 Glossary

The following glossary is used in this document:

Archive Facility (Site), Archive Cluster	Refers to an NGAS based archive system within an organization, where all data produced are being managed and from where online access to the data and processing facilities are provided. There might be several such Archive Facilities within an organization.
Archive Request	Request from a client of the NG/AMS Server to have a file archived.
Back-Log Buffering	Back-Log Buffering can be carried out by the NG/AMS Server if an error occurs, which makes it impossible to archive the file at that moment. The file will thus be stored temporarily in the Back-Log Buffer Area. The NG/AMS Server (Janitor Thread) will attempt at a later stage to handle the file.
Bad File	A Bad File is a file that could not be accepted for archiving by NG/AMS. I.e. it was rejected by the DAPI handling this file type. This could e.g. be due to a wrong expected size of a FITS file.
Bad Files Directory, Bad Files Area	Area on the disk where files, which are mal-formed are stored. There is a Global Bad File Directory on one of the system disks on each NGAS Node. Apart from that, there is a Bad Files Directory on each archive disk installed.
Data Provider	An NG/AMS Server to which one or more Subscribers have submitted a request for data. Each time a new file becomes available on this NG/AMS system, the Data Provider will check if it should be delivered to one or more of its Subscribers.
Data Subscriber	Client that has subscribed itself to receive a certain kind of data from an NG/AMS Server. In order for a Subscriber to subscribe itself it must send the SUBSCRIBE Command. In order to un-subscribe, it must send the UNSUBSCRIBE Command.
Disk, Hard Disk Drive, HDD	In the context of NG/AMS the term disk refers to a random access storage device, which can be mounted under UNIX and which has a file system created on it.
Disk Dictionary	A dictionary containing information for the Storage Media available in an NGAS Host.
Disk Set	A set consisting of one or two Storage Media; see also Storage Set.
Dynamic Disk Set	A temporary association between two non-completed Storage Media. This association exists as long as NG/AMS is Online. I.e., there is no static link between NGAS Storage Media.
Logical Name	A 'human' readable name that is used when referring to disks. The disks will typically be labeled with the Logical Name. Should be unique, although this may not be guaranteed. Note, the Disk ID must be uniquely allocated per disk.
Main (Data) File	The copy of the data file stored on the Main Storage Area.
Main Disk	Data is archived onto Storage Sets. The Main Disk is the primary Storage Media of the set and must also be present.
Main (Storage) Area	The array of HDDs in an NGAS Node, which are sent to the Archive Facility Site when filled with data.
NG/AMS Server	The central process of NGAS, which executes all the actions necessary to archive, retrieve, process and maintain and handle the archived data.
NGAS Cluster	Refers to a set of NGAS Hosts, which are used together as an archive unit.
NGAS Configuration	The XML based document used as input to the NG/AMS Server when this is started. Contains all the information needed for NG/AMS to operate in a given context. Note, the configuration can be contained also in the NGAS DB. This is used for the operational systems from V2.3.
NGAS Host/NGAS Node	Refers to a WS which has the NG/AMS SW installed and which is used as an archiving or data server unit.
Physical Disk Dictionary	Dictionary that contains information about each NGAS Storage Media available

	in a certain NGAS Node.
Processing Area	Directory used to store temporary copies of files to be processed and other temporary files created during processing.
Production Site, *Data Production Site*	The site, e.g. at the telescope site, where data is produced and being archived into an NGAS System.
Replication (Data) File	The copy of a data file, which is stored in the Replication Area.
Replication (Storage) Area	The array of HDDs that contains the replicas of the data on the disks in the Main Storage Area.
Staging Area	A storage location (directory) used to temporarily store data files being handled. NG/AMS e.g., uses a Staging Area on each Target Disk, for receiving data files before moving the files to their final location.
Storage Media	Refers to a storage unit used in the context of NGAS for receiving data being archived and from which archived data is retrieved. Used interchangeably with Storage Disk (disk), as HDD's is the media used presently to store data.
Storage Set	A storage unit, which consists of either one or two disks on which data is archived.
Subscriber	See Data Subscriber.
Subscription Back-Log	A buffer in which data files that could not be delivered to Subscribers are stored until a delivery can be carried out successfully.
Target Disk/Target Storage Media	Media selected to receive a file being archived.

XML Dictionary Key Format	A simple scheme for mapping XML components into a dictionary key (hash key). This is used e.g. to shred an XML document for storage in a dictionary or hash table or in the DB. For instance, a key defined as:
	
	`NgamsCfg.Log[1].LocalLogLevel`
	
	- maps into the following XML element:
	
	`<NgamsCfg>`
	`  <Log LocalLogLevel="<value>"/>`
	`</NgamsCfg>`
	
	The mapping between the two formats is 1:1.

Table 4: Glossary used in this manual.

2. Overview

The main objectives of NG/AMS are:

> *To provide an archive system with all services needed for archiving data in a safe and efficient manner. Furthermore to provide transparent access to the data in the archive holding, combined with processing of the data if requested. Finally, the purpose is to provide tools and services for the operators of the system, which facilitate the task of maintaining the data in a good condition and carrying out the necessary handling of the archive facility.*

In this chapter the basic concepts of NGAS and NG/AMS are described. An overview of the NG/AMS is given as well as a description of the various fundamental features and services provided by NG/AMS. This chapter provides a somewhat high-level description of the most important features and services. More in-depths descriptions can be found in the following chapters.

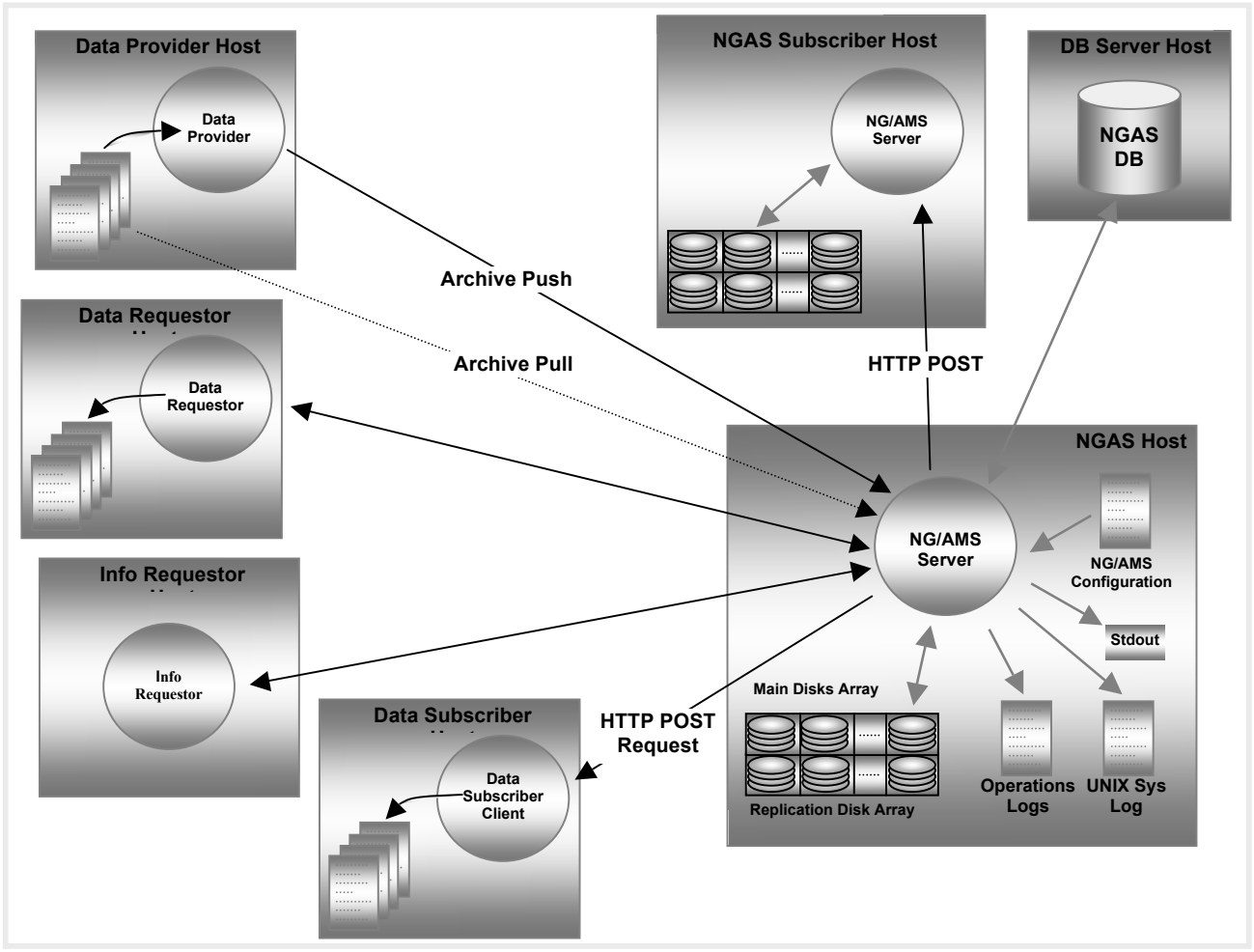

Figure 1: Example operational environment of the NG/AMS Server.

1.8 The Concept of NGAS & NG/AMS

The concept of NGAS is to use random access storage media to obtain high I/O performance. However, any device that can be mounted under UNIX (Linux) and on which a file system can be created, can become an NGAS Storage Media.

Some of the basic advantages of the NGAS concept are:

- The archiving of data files can be carried out very fast. The archiving performance (throughput) can be scaled up easily according to the needs, by adding more archiving units.

- Data is online as soon as it has been archived.
- It is not necessary to store data in an intermediate location and to generate later the final media. The final media is created 'real-time'.
- The processing power of the computers hosting the disks can be used to process the data both during archiving and while retrieving data.
- It is possible to archive data remotely via the HTTP protocol.
- The price per storage unit is relatively low compared to other solutions.

The NGAS is based on standard PCs HW running Linux. These NGAS Nodes are normally equipped with a set of HDD sliders in which HDDs can be inserted and removed easily.

It is foreseen to have one or more NGAS Hosts at the telescope sites, one in connection with each major producer of data. As soon as a disk is full, it will be send to the Archive Facility Site where it will be installed in a free slot in an NGAS Node in the archive NGAS Cluster. The data is immediately online as soon as the NG/AMS has 'recognized' the disk. NG/AMS can produce a Replication Disk so that a back-up of the data is available. Although the system has been developed on Linux (UNIX) it may be possible to port the SW relatively easily to other platforms supporting Python like e.g. MS-Windows or MAC-OS.

The philosophy behind the NG/AMS SW is to provide an open architecture that can be extended and adapted to be used as a generic archive facility in many different contexts. Therefore, the NG/AMS SW itself does not have any specific functionality built-in to handle specific types of data or specific HW. All this 'knowledge' must be implemented and made available for an NG/AMS Server in order to make it carry out the requested tasks. This is done by providing a set of specific services in the form of plug-ins, which are simple Python functions. Due to this scheme, it is possible to adapt NG/AMS with a minimum amount of effort to handle e.g. many different kinds of data.

The heart of the NG/AMS is the NG/AMS Server. This is a multithreaded server based on the standard HTTP protocol. It can be seen as a dedicated WEB server. Since the server is multithreaded it is possible to issue several requests simultaneously[1]. A number of commands are provided by NG/AMS. For more detailed information about these commands consult chapter 16. For more information about the technical details of the command interface, consult chapter 7.

1.9 Services & Features

Some of the main services and features provided by the NG/AMS SW are:

Access/Service Restriction	It is possible to enable/disable some basic services via the configuration. The services in questions are for the moment: 1) Handling of Archive Requests, 2) Handling of Retrieve Requests, 3) Data Processing and 4) Remove Requests (removing of disk and file information from the system) (see section 1.25). In addition, user access can be authorized based on the HTTP authentication protocol supported by NG/AMS.
Adding of Specific Behavior Based on Plug-In Concept	NG/AMS is implemented in a way so that only the kernel/general functionality is implemented (hard-coded) into the server SW. All the context specific features are provided based on a plug-in scheme making it possible to adapt the server in a very flexible way. As an example of this, the specific handling of data during archiving, is done by a plug-in provided for each type of data (see the chapters 1.58-1.67).
APIs for C and Python	APIs for communicating with the server are provided for applications written in C and Python (see the chapters 9 and 10).
Back-Log Buffering of Data	In case problems occur preventing NG/AMS from archiving data, NG/AMS will Back-Log Buffer data and try to handle this at a later stage (see section 1.21).
Cache Archive Service	Running a special instance of the NG/AMS Server, it is possible to operate an NGAS Unit as a cache archive, whereby files are kept in the archive for a certain amount of time, defined by various criteria. The Cache Control Plug-In is used to control when to remove files from the cache archive.
Canalization of Data Streams	Via the configuration file it is possible to define how NG/AMS should stream data onto the various Storage Disks available in an NGAS Host (see section 1.13).
Cluster Mode	NG/AMS is prepared for operation of a set of NGAS Nodes in a cluster that constitutes an archive data server and processing facility (see section 1.28).
Command Line Utilities	Two command line utilities for communicating with the NG/AMS Server are provided.

[1] A limiting factor of the present implementation, is that the Python "threading" module is used to provide multithreading. This module in reality does not utilize all CPU cores on a multicore system. To overcome this limitation, multiple NG/AMS Server may be executed on each node.

	These are based on the NG/AMS C and Python APIs (see section 1.41).
Data Consistency Checking	If enabled, an NG/AMS Server will run a periodic data consistency check of the data stored on the disks under its control. Via a number of parameters it is possible to adjust quite accurately how much load and how long time this task should take up (see section 1.23).
Data File Archiving via Push/Pull Technique	Efficient archiving of data files is provided based on an Archive Pull Technique, whereby NG/AMS picks up files given by a URI, and on an Archive Push Technique, where the data provider writes (pushes) the data to the server (see section 1.15).
Data File Retrieval & Processing	NG/AMS provides a scheme for transparent access to the data. Based on the information in the NGAS DB, a contacted NG/AMS Server can locate the data requested by the user and provide this to the user by acting as a proxy (transparent data access). It can also send back HTTP redirection messages to indicate to the data requestor where to find the data. The C and Python APIs handle the data access completely transparent for the client (see section 1.16).
Data Replication	NG/AMS can handle replication of data files if requested. Also the information for such replicated files is updated automatically in the NGAS DB (see the section 1.15 and chapter 1.61).
Data Subscription Service	A service is provided to export data being archived to Subscribers with an interest in the data or a sub-set of this. A Subscriber can either subscribe to all data being archived on an NGAS Host, or to part of it. Latter is done by means of the Filter Plug-In that is applied on the data to determine whether to export it or not. In this way, it is e.g. possible to synchronize data holdings between different NGAS Nodes (see section 1.29).
Disk Registration & Supervision	When a disk first has been registered by NG/AMS, the movements of the disk will be monitored by NG/AMS, so that when it appears in an NGAS Host the NGAS DB will be updated to indicate the latest status of the disk (see section 1.14).
Email Notification Service	A service is provided for notifying subscribers about various events occurring during operation. Examples of such events are errors, disk change requests and data inconsistency reports (see the sections 1.18 and 1.19).
Extendable for Usage with Various DBMS'	NG/AMS is prepared for usage with various DBMS'. For now only Sybase is supported, but this can easily be expanded.
File Cloning	A service is provided with which it is possible to clone single files, sets of files, or entire disks (see section 1.82).
File Registration	A number of parameters are registered for the files archived in the NGAS DB, making it possible to locate, retrieve and process these files.
Flexible Adaptation via Configuration	The NG/AMS Server is configuring itself at start-up, based on a large number of configuration parameters defined in the NG/AMS Configuration, which is an XML document. The configuration may also be contained in the DB. The advanced configuration makes it possible to adapt the server for specific contexts in a flexible way (see chapter 6).
Generation of Checksum	NG/AMS generates a checksum value for each file generated. This is based on a plug-in concept so that context/data specific checksum calculation can be applied (see chapter 1.64).
HTTP Protocol	The communication interface of NG/AMS is based on the standard HTTP protocol. This makes it easy to access the server from various clients. It is even possible to interact with the server using a WEB browser (see section 1.22).
Information Query	A set of various types of information can be queried via the STATUS Command. This information is such as the state of the system, or information about files and disks (see section 0).
Logging	A quite substantial set of information can be logged according to different levels: 1) On stdout, 2) In the UNIX syslog, and 3) In a log file (see section 1.17).
Mirroring Service	The NG/AMS Server comes with a built-in mechanism to mirror data cross RDBM boundaries. This makes it possible to synchronize independent NGAS Clusters, via an HTTP channel.
Multithreaded Server	The NG/AMS Server is using threads when handling requests from clients. This means that it is capable of handling several requests simultaneously.
Production of Disk Labels	NG/AMS can produce labels for the disk cases on request. The actual SW to operate the printer must be provided in the form of a plug-in (see section 1.24).
Removing File and Disk Information	Two commands are provided to remove single files or set of files. Another command is provide to remove an entire disk from the system (see the sections 1.93 and 1.94). In addition, a command, DISCARD, is provided to suppress files in the archive (section 1.84).
Simulation Mode	NG/AMS provides a Simulation Mode, which makes it possible to operate the system

	without the availability of the actual HW, like the disk controller, disks, etc. Running in Simulation Mode, a simulated NG/AMS environment is generated on a single disk. This is useful for test and development. The Simulation Mode however, could also be used to run an NG/AMS on a 'normal' workstation for archiving data in a production system (see section 1.20).
State Management	The NG/AMS Server maintains a State/Sub-State scheme to make it possible to restrict the services provided according to the 'condition' of the server (see section 1.11).
Suspension/Wake-Up Service	An NG/AMS Server can be configured to suspend the NGAS Host where it is running. A service is provided so that another NGAS Server can be requested to wake up an NG/AMS Server suspending itself (see section 1.30).
SW Modularity/Re-usage	The NG/AMS SW is implemented as a number of classes and library functions, which can be used to build dedicated servers and other applications if needed (see chapter 13).
Thorough Documentation	Apart from this manual, thorough and accurate documentation contained in the Python source code of NG/AMS is provided. This makes it possible to browse the documentation online e.g. using “pydoc”[2] (see section 1.76).
User Command Plug-ins	It is possible to enhance the command interface with new commands, added per plug-in basis.
User Service Thread	The User Service Thread is invoked periodically in background. It can be used to carry out various tasks to be handled to e.g. keep the system tidy.
XML Information Exchange	All information sent back from the server (status messages) are based on XML (see e.g. chapter 12).

The services and features listed above and described shortly, are explained in more detail in this and the following chapters.

1.10 Starting & Stopping the NG/AMS Server

The NG/AMS Server must be invoked with a number of different command line parameters. These are described in section 1.40. It is mandatory to specify an NG/AMS Configuration to be used by the NG/AMS session. The configuration can be contained in the NGAS DB. How to configure the NG/AMS environment is described in chapter 6. The server can be started with the "-v" option to produce output on “stdout”. Normally, in a production environment, it will be started as a back-ground process, which only produces log output to the UNIX “syslog” and/or a Local Log File (see also section 1.17).

The server can be stopped either by sending a “SIGTERM” signal (15) or by sending an EXIT Command, which can be issued when the server is in Offline State (see also 1.11). If the server is killed with a “SIGTERM” signal, it will invoke internally a signal handler that cleans up the environment and shuts it down in a proper manner whereby also the System Offline Plug-In (chapter 1.59) is invoked. Also when issuing an EXIT Command, the server invokes the proper 'clean-up procedure'. If the server is killed by a “SIGKILL” (9) signal, the signal handler is not invoked, and the server leaves its environment in an 'undefined' state. This also happens if the computer on which the server is running is shut down abruptly. If this happens it will be necessary to start the server subsequently with the "-force" parameter to force it to start-up. It is possible to ‘clean up’ the environment by bringing the server Online/Offline in the proper manner.

[2] http://www.python.org/doc/current/lib/module-pydoc.html

1.11 The NG/AMS Server States & Sub-States

The NG/AMS Server is maintaining a scheme of a State and a Sub-State that determine which services the server can handle at a given point in time and which indicate the 'condition' of the server. The States and Sub-States and the corresponding conditions are as follows:

State ò Sub-State ð	*Idle*	*Busy*
Offline	This is the condition in which the NG/AMS Server enters after starting up, and when the OFFLINE Command has been issued. In this state only the STATUS Command is accepted. I.e., no Archive or Retrieval Requests are handled. The EXIT Command is also accepted. Latter makes the server clean up and terminate.	In this state the server is performing the transition from Offline to Online, or is preparing to exit from execution. No commands are accepted.
Online	In this state the server is ready to handle commands like ARCHIVE and RETRIEVE. In addition the OFFLINE Command is accepted.	In this state the server is busy handling one or more Archiving or Data Retrieval Requests. Also the STATUS Command is accepted. An OFFLINE Command will be rejected.

Table 5: NG/AMS State/Sub-States.

It is possible to query the state of the server by issuing a STATUS Command without parameters. The reply to a STATUS Command is an XML document with the following contents:

```
<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
  <Status Date="2002-12-23T13:15:52.194" HostId="acngast1" Message="Successfully handled command STATUS"
          State="ONLINE" Status="SUCCESS" SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
</NgamsStatus>
```

Figure 2: Response to STATUS Command.

1.12 The NG/AMS Storage Media Infrastructure

The Storage Media infrastructure used by NG/AMS is depicted in ***Figure 3***.

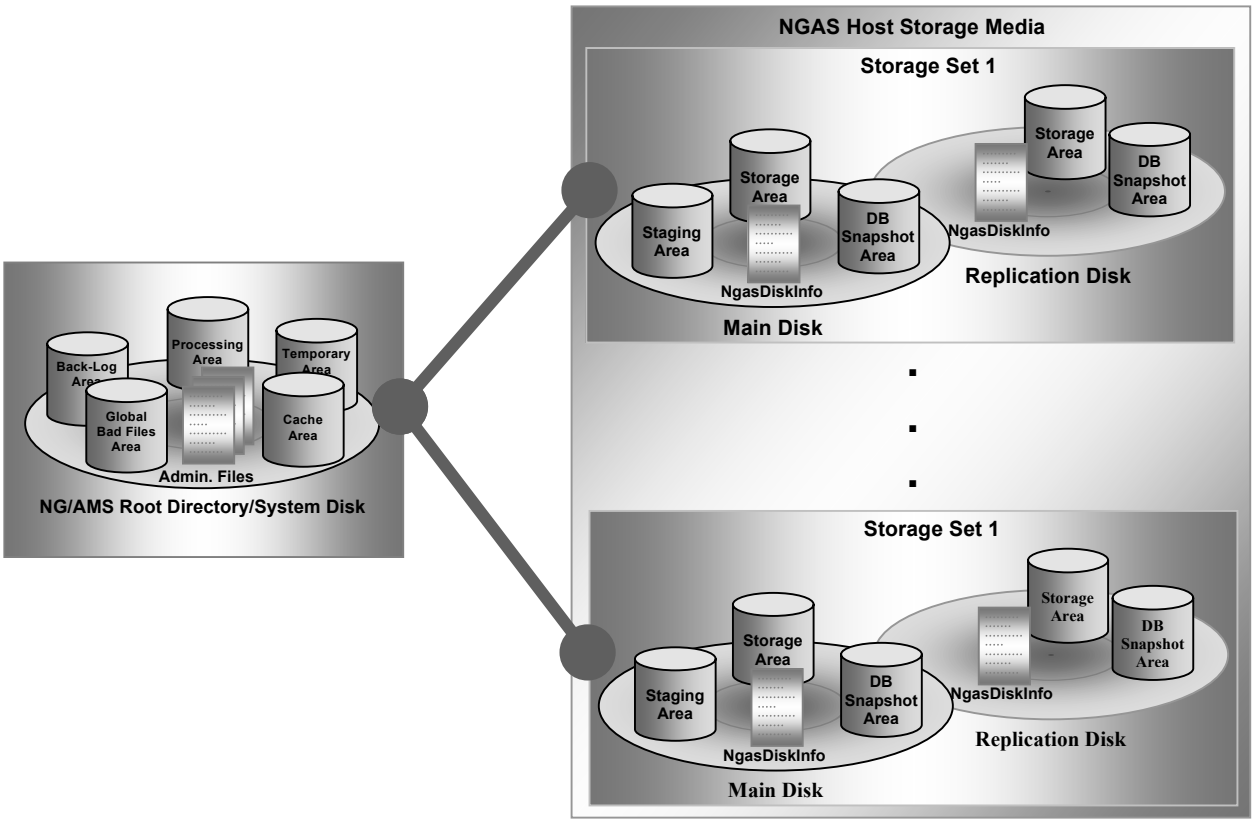

Figure 3 The NG/AMS Storage Media Infrastructure.

During operation (archiving), disks are usually used in pairs as shown in ***Figure 3***. This association of disks however, is handled in a dynamic manner, so that disks are associated with each-other when NG/AMS goes online and as long as it remains Online. This dynamic association is done, based on the definition of Storage Sets in the NG/AMS Configuration (see section 1.46, XML element: “StorageSet”). As soon as the server goes Offline, the association does not exist anymore.

	*In the context of NGAS, data is seen file-wise and not disk-wise and no attempts are made in order to maintain identical sets of files on different disks that have been associated during operation at a certain point in time.*

It is possible to have a Disk Set consisting of only a Main Disk. It is recommended however, always to use the Replication Service of NG/AMS when relevant, to increase data safety.

A safety check is implemented in the SW such that if a Storage Set is defined to have two disks and if only one of the two disks is available (present or not completed), that Storage Set is rejected.

A Storage Set is considered to be completed when either of the two disks in it are considered as ‘full’ (see also section 1.19). If a Storage Set consists only on one Main Disk, it is considered completed when the Main Disk is full. Beware, that since the association between disks only exists during operation, it may be that only one of the disks in a set is actually marked as completed in the DB, whereas the other remains un-completed and can be used together with another disk. In this way it is e.g. possible to use Main Disks of size 80 GB together with Replication Disks of size 200 GB. In the definition of the Storage Set in the NG/AMS Configuration, it is possible to ‘lock’ two disks together, so that when one of the disks is completed, also the other disk will be marked in the DB as completed (CFG: “NgamsCfg.StorageSet:Synchronize”). Synchronization should normally be used when disks of the same size are used together, to avoid that one disk remains un-completed whereas in reality hardly any space is available on the media.

As seen in ***Figure 3***, the NG/AMS Storage Media infrastructure, is based on a single root directory under which the Storage Disks are mounted. Under this area, NG/AMS is also storing some files for internal purposes. Among these is a file containing the PID of the NG/AMS Server process (FILE: "<NgamsCfg.Server:MountRootDirectory>/.NGAS_<NGAS ID>", e.g. "/NGAS/.NGAS-pa12-7777"). The Back-Log and Global Bad Files Directories can be placed in a location of choice. This is done via the NG/AMS Configuration. The names of these directories are "<NgamsCfg.ArchiveHandling:BackLogBufferDirectory>/bad-files" and "<NgamsCfg.ArchiveHandling:BackLogBufferDirectory>/back-log".

Also located under the NGAS Mount Root Mount, are the Temporary and Cache Storage Areas. Former is used for storing temporary files during processing. These will be deleted after the processing terminates or after a given time-out during which no access to these temporary files was attempted. Latter, is used to store files, which are kept between two sessions of executing the NG/AMS Server.

The Processing Area (Directory) shown in ***Figure 3***, is used by NG/AMS for storing temporary files while doing file processing. The files stored in this directory will be removed by NG/AMS after the processing has finished. The name of this directory is: "<NgamsCfg.Processing:ProcessingDirectory>/processing". Some care should be applied when determining the location of these directories, since it may have an influence on the performance of the system. E.g., if a location for the Processing Area is chosen, which has a poor I/O performance; this may slow down the processing considerably.

It is possible to make NG/AMS carry out replication of the files being archived. This feature can also be disabled (CFG: "<NgamsCfg.Server:Replication>").

The data volumes must be reachable via the paths given by the Root Directory and Volume Directory (CFG: "<NgamsCfg.Server:RootDirectory>" and "<NgamsCfg.Server:VolumeDirectory>").

The data files archived will be stored under a single directory (referred to as Storage Area in ***Figure 3***) in the mount directory on the target disks. The name of this area is configurable (CFG: "<NgamsCfg.FileHandling:PathPrefix>"). It is up to the DAPI implementation to define the structure of the directories and files within the Storage Area. On the data disks there is also a Staging Area used by NG/AMS when receiving data files. Data is received directly onto the Main Target Disk for efficiency reasons. The name of this directory is: "<Volume Directory>/<disk mount directory>/staging". There is only one such Staging Area on the Main Disk. Also located on the Data Disks is a file named "NgasDiskInfo". This file is an XML document that contains a summary of the information about the disk contained in the DB. An example of such a file can be found in section 1.73.

On each disk, the DB Snapshot Area is contained. It is used to keep track of the files registered in the NGAS DB for the given disk. This is used to synchronize remote NGAS DBs in case changes are introduced in the data stored on the disk. For further information see chapter 1.52.

1.13 Data Classification & Handling

One of the fundamental concepts behind NG/AMS is the way data is classified and handled. This is based on the same concept as used by many WEB browsers and mail tools, namely on the mime-type of the data, which again is derived from the extension of the data files. It is also possible to explicitly specify a mime-type for a data file when issuing it for archiving. In NG/AMS no mime-types for the data files handled are hard-coded into the SW. By means of the NG/AMS Configuration, mime-types for new types of data files to be handled can be added. Note that for new types of data a corresponding DAPI must be provided (see chapter 1.61). If NG/AMS encounters a data file with an unknown mime-type (not defined in the configuration) while handling an Archive Request, the request will be rejected. Such data files will not be buffered on the NGAS Host handling the request.

It is also possible to define an arbitrary number of Data Streams, normally one per each type of data to be handled. In the data stream the following information must be defined:

Mime-Type	The mime-type indicating for which data the stream is defined.
DAPI + Parameters	The DAPI that should be used to handle the processing and archiving of the data file. In addition parameters for the DAPI can be specified in the configuration file.
Target Storage Set(s)	One or more Storage Sets, on which the data can be stored.

Table 6: Parameters for data classification.

See chapter 6 for more information about the NG/AMS Configuration.

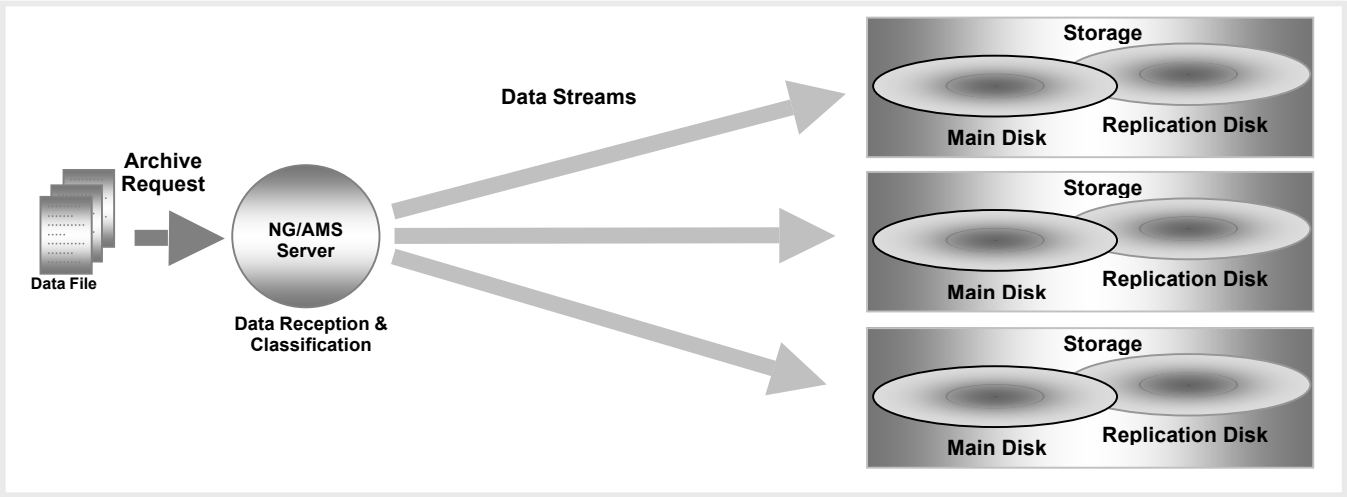

Figure 4: Data channeling.

Two standard mime-type are used by NG/AMS. These are:

text/xml	Used by NG/AMS to indicate that a reply contains an XML document.
ngas/archive-request	Generic mime-type used to indicate for NG/AMS that it should determine the mime-type from the file URI. It is also possible to specify the mime-type explicitly in an Archive Request.

Table 7: Reserved mime-types.

Using the NG/AMS APIs (see the chapters 9 and 10) the user/client normally does not have to worry about this aspect.

1.14 Disk Handling/Life Cycle of an NGAS Storage Media

In this section the various stages in the life cycle of an 'NGAS disk' are described. In the diagram in ***Figure 5***, a typical life cycle for an NGAS Storage Media is shown. There might be differences for the various contexts how the actual disk handling is implemented.

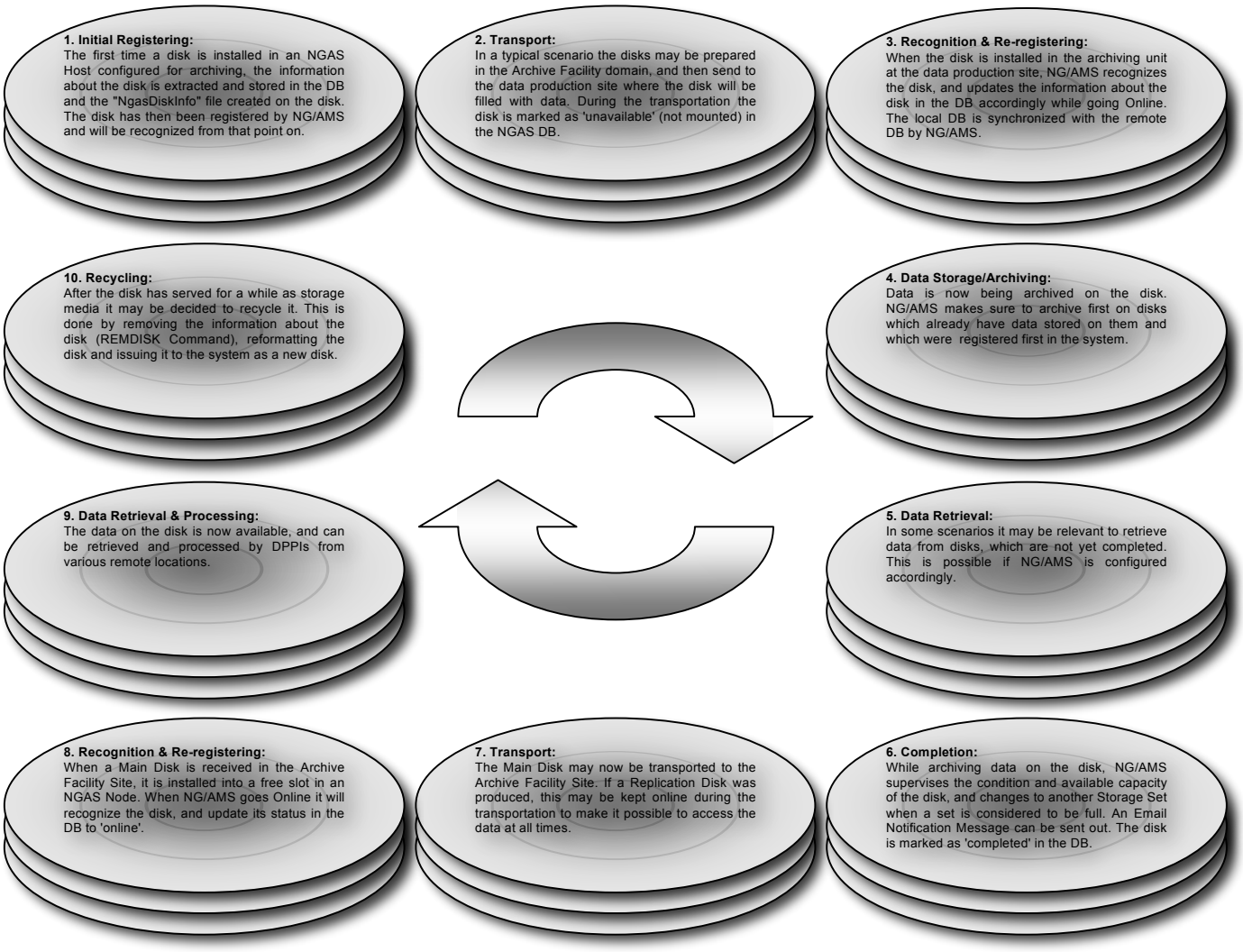

Figure 5: Life cycle of an NGAS disk.

Empty NGAS disks, used for archiving purposes, are usually kept together in pairs. NG/AMS however, does not impose a static association between disks, and the association of non-completed disks. The provided Logical Name serves mainly to make the operation of an NGAS based archive system more convenient. For more information about the handling of disk association consult section 1.12.

3. Basic Features

In this chapter the basic features from the list in section 1.9 of NG/AMS are described in detail.

1.15 Data File Archiving

Archiving of files in NG/AMS is done via the ARCHIVE Command. Data files can be archived either by using the Archive Push Technique or the Archive Pull Technique. Using the former, the application archiving the file, reads in the file and sends the contents of the file in the HTTP message body. When making use of the latter technique, the client sends a URL indicating a location where the file can be picked up by NG/AMS. This URL must therefore be accessible for NG/AMS.

NG/AMS maintains a scheme of file versioning, whereby if a file with the same File ID is issued several times, the version number in the NGAS DB (DB: "ngas_files.file_version") is incremented by one. The first file with a given ID has version number 1.

In short, the handling of an Archive Request is done as follows:

- NG/AMS receives the Archive Push or Pull Request.
- It determines the mime-type of the data file. From the configuration and the current disk status, which it reads from the NGAS DB, it finds a suitable Target Disk for the data.
- The data is subsequently received into the Staging Area of the Main Target Disk.
- Afterwards the DAPI corresponding to the type of data in question is invoked to do the specific handling of the data.
- After the DAPI has finished its processing, it returns control to NG/AMS.
- If a Data Checksum Plug-In is specified, this is invoked by NG/AMS to generate the checksum for the archived file (see 1.64).
- The DAPI has then extracted/produced the necessary information for NG/AMS to be able to update its status in the DB and to be able to move the file to its final destination.
- If replication is enabled, NG/AMS replicates the file and updates the information for the Replication File in the DB.

It is possible specify a mime-type explicitly when issuing an Archive Request. In this case NG/AMS will not try to resolve the mime-type according to the Mime-type Mapping defined in the configuration. When doing this, the mime-type will be taken as it is and the file handled according to this. It is still necessary to define a Stream for such a client specified mime-type in the configuration as otherwise NG/AMS would not know how to handle the file. Using a client specified mime-type, it is possible to archive a file with a mime-type already defined in the configuration, such that the file is processed with a different DAPI.

In connection with the Archive and Clone Request handled in parallel, it is important that the threshold defined in the configuration, for considering a disk completed, is defined as 2 times the total sum of the size of the data of the maximum possible number of simultaneous Archive Requests.

The ARCHIVE Command is described in section 1.79. A more detailed description of the procedure executed by NG/AMS while handling an Archive Request can be found in chapter 1.61 (describing the DAPI).

1.16 Data File Retrieval & Processing

Archived files can be retrieved from NG/AMS using the RETRIEVE Command. In the present implementation it is only possible to retrieve one file at a time. It is possible to request to have the data processed by a DPPI before NG/AMS returns a reply. The concept of the DPPIs is described in detail in chapter 1.63.

When NG/AMS receives a Retrieve Request, it checks the NGAS DB for the various instances of a data file with the given ID, which is online. Several versions may be available. The decision of which file to choose, is done as follows:

- A list of all files with the given File ID, which are registered as being online, is retrieved from the NGAS DB. In addition, files marked to be 'ignored' are not considered.
- The files are ordered according to their File Version, whereby the latest file (highest version number), gets the highest priority.
- It is subsequently investigated where the instances of the given file are located. Four cases are considered:
  - **Local Host**: The file is store don the local host.
  - **Cluster**: The file is stored on a node within the same cluster as the node handling the Retrieve Request.

- **Domain**: The file is stored on a node which is connected directly to the same domain and the node handling the request, or on a node controlled by a master unit, which has the same domain.
- **Remote**: The data item is stored on any NGAS Node connected to a remote network.

I.e., if a file with a given ID and the same version is found on several NGAS Hosts, the selection criteria for which file to take is done according to the priority list described above. I.e., if a file is found on the local host and on a remote host, the instance on the local host is taken.

- When a file is found to be residing on the local host, NG/AMS always gets the file, processes it if requested and sends it back to the requestor. If the file is located on a remote host, NG/AMS will either send back an HTTP Redirection Response (HTTP Status Code: 303), or it can act as proxy. In the former case it is up to the requestor to re-issue the request to the NGAS Host referenced in the HTTP Redirection Response (see also section 1.50). In the latter case, it retrieves the file from the remote location, and subsequently sends back this file to the requestor. Whether an NG/AMS Server should act as a proxy or not, is configurable (CFG: "NgamsCfg.Server.-ForceProxyMode", 1 = Proxy Mode).

	*If a file is located on a host within the private network or on a remote host, possible processing requested will be carried out on the host on which the file resides*.

Methods for retrieving files in an easy manner are provided by the C and Python APIs. See the chapters 9 and 10.

1.17 Logging

A number of different types of log output can be produced by NG/AMS. These and their properties are:

Log Type	*Description*
Local Log File	The location of a Local Log File is defined in the NG/AMS Configuration file (CFG: "NgamsCfg.Log:LocalLogFile"). I.e., the user can decide himself where to put this file. The level (intensity) with which there is logged, can be adjusted as well (CFG: "NgamsCfg.Log:LocalLogLevel"). Note, that NG/AMS will continue to write (append logs) in the same log file. I.e., it should be considered to implement means to purge the log file[3] periodically. The format of the Local Log file log entries is as follows: <ISO 8601 time stamp> **[**<type>**]** <log message> **[**<source file>**:**<method>**:**<line>**:**<PID>**:**<thread>**]**[4] where <type> is defined as: <type> := EMERGENCY | ALERT | CRITICAL | ERROR | WARNING | NOTICE | INFO | DEBUG Examples of some entries in a Local Log File are: … 2004-08-10T09:04:42.095 [INFO] Handling HTTP request: client_address=('134.171.21.31', 35295) - method=POST - path=|ARCHIVE| - user-agent=NG/AMS C-API - content-disposition=attachment; filename="/opsw/packages/ngams/ngamsTest/src/TinyTestFile.fits"; no_versioning="0"; wait="1" - content-type=ngas/archive-request - content-length=2880 [ngamsServer.py:handleHttpRequest:-1180:1678:Thread-1] 2004-08-10T09:04:42.148 [INFO] Received command: ARCHIVE [ngamsCmdHandling.py:cmdHandler:43:-1678:Thread-1] 2004-08-10T09:04:43.937 [INFO] NGAMS_INFO_FILE_ARCHIVED:4027:INFO: Successfully archived file with URI: TinyTestFile.fits. Time: 0.756s [ngamsArchiveUtils.py:dataHandler:765:1678:Thread-1] …
(UNIX) Syslog	It is possible to instruct NG/AMS to produce log entries into the UNIX syslog. This is only done when certain important events occur. Such events are error conditions, and handling of archive requests. Whether or not to log into syslog is specified in the configuration file (CFG: "NgamsCfg.Log:SysLog"). It is possible to specify an ID, which is written in each log entry in the syslog (CFG: "NgamsCfg.Log:SysLogPrefix"). This makes it possible to filter out logs for a certain context at a later stage. An example of some syslog entries produced by NG/AMS is: … Feb 20 12:58:04 w2p2nbu python: DFSLog:2002-02-20T12:58:04.200 Error w2p2nbu NGAMS_ER_DISK_INACCESSIBLE:3004:ERROR: Disk with ID: Slot ID: 3 - Disk ID: IC35L080AVVA07-0-VNC400A4C1G8RA is not accessible (writable). Feb 20 12:58:04 w2p2nbu python: DFSLog:2002-02-20T12:58:04.410 Error w2p2nbu NGAMS_ER_DISK_INACCESSIBLE:3004:ERROR: Disk with ID: Slot ID: 4 - Disk ID: IC35L080AVVA07-0-VNC400A4G0KZ8A is not accessible (writable). Feb 21 23:43:56 w2p2nbu python: DFSLog:2002-02-21T23:43:56.800 Notice w2p2nbu Disk with ID: IC35L080AVVA07-0-VNC400A4C1607A - Name: LS-FitsStorage3-M-000027 - Slot No.: 5 - running low in available space (4938 MB)! Feb 22 09:29:40 w2p2nbu python: DFSLog:2002-02-22T09:29:40.740 Notice w2p2nbu Marked Main Disk with ID:

[3] A mechanism for this may be provided later within NG/AMS.
[4] Characters in bold are part of the actual contents.

	IC35L080AVVA07-0-VNC400A4C1607A - Name: LS-FitsStorage3-M-000027 - Slot No.: 5 - as 'completed' - PLEASE CHANGE! Feb 22 09:29:40 w2p2nbu python: DFSLog:2002-02-22T09:29:40.770 Notice w2p2nbu Marked Replication Disk with ID: IC35L080AVVA07-0-VNC400A4C1622A - Name: LS-FitsStorage3-R-000027 - Slot No.: 6 - as 'completed' - PLEASE CHANGE! …
Verbose Log	The Verbose Logs are written to stdout. They contain more detailed information than the two other types of logs. This type of log is usually used for debugging, trouble shooting and test purposes. The Verbose Log Level is adjusted via a command line parameter (-v <level). If this parameter is not specified, no Verbose Log output is produced. The format of the Verbose Logs is as follows: <ISO 8601 time stamp>:<module>:<method>:<line no.>:<PID>:<thread name>:<log type>**>** <log message> The various values for <type> are defined in connection with the Local Log File. An example of Verbose Log output is (lowest Log Level (=1)): … 2004-08-10T09:04:42.095:ngamsServer.py:handleHttpRequest:1180:1678:Thread-1:INFO> Handling HTTP request: client_address=('134.171.21.31', 35295) - method=POST - path=|ARCHIVE| - user-agent=NG/AMS C-API - content-disposition=attachment; filename="/opsw/packages/ngams/ngamsTest/src/TinyTestFile.fits"; no_versioning="0"; wait="1" - content-type=ngas/archive-request - content-length=2880 2004-08-10T09:04:42.148:ngamsCmdHandling.py:cmdHandler:43:1678:Thread-1:INFO> Received command: ARCHIVE 2004-08-10T09:04:42.150:ngamsArchiveCmd.py:handleCmdArchive:80:1678:Thread-1:INFO> Handling Archive Push Request ... 2004-08-10T09:04:42.181:ngamsArchiveUtils.py:dataHandler:664:1678:Thread-1:INFO> Archiving file: TinyTestFile.fits with mime-type: image/x-fits ... 2004-08-10T09:04:42.349:ngasGarArchFitsDapi.py:ngasGarArchFitsDapi:177:1678:Thread-1:INFO> DAPI handling data for file with URI: TinyTestFile.fits 2004-08-10T09:04:43.937:ngamsArchiveUtils.py:dataHandler:765:1678:Thread-1:INFO> NGAMS_INFO_FILE_ARCHIVED:4027:INFO: Successfully archived file with URI: TinyTestFile.fits. Time: 0.756s …

Table 8: The supported log output formats.

The Log Level is a number in the range from 1 to 5, whereby 1 is the 'high-level' logs and 5 is the lowest (deepest) level, providing the most detailed information. The interpretation of the various Log Levels is as follows:

Level	*Description*
1	The lowest Log Level, which only provides a brief summary of the actions performed. Errors and warnings are always logged.
2	This level provides more thorough information about the actions performed.
3	This level performs a quite extensive set of logs describing in details the various actions carried out by NG/AMS and the plug-ins invoked by this. For logging in the log file, there should normally not be logged with a higher level than 3.
4	This level provides a very profound set of information. It is usually only used for debugging and test purposes and for locating errors occurring in the system. Logs may be produced cyclically from the Data Consistency Checking and Janitor Threads.
5	The deepest level provides a quite extensive set of logs. Some of the log will be quite repetitive. The quantity of log information produced is quite big, and if logging into a log file with this level, care should be taken that it may grow in size very rapidly.

Table 9: Interpretation of Log Levels.

The level (intensity) with which there should be logged as well as name of Local Log File and a prefix for the syslog entries, can be specified in the NG/AMS Configuration (CFG: "NgamsCfg.Log"). For further information about this specific properties see chapter 6.

For the logging it is possible to specify the name of the log file (CFG: “NgamsCfg.Log:LocalLogFile”). It should also be specified with which level there should be logged into the log file, normally a level of 3 is adequate for a normal operational environment (CFG: “NgamsCfg.Log:LocalLogLevel”). The logging system used by NG/AMS provides a scheme for buffering logs. The amount of log entries, i.e., the size of the log cache must be defined (CFG: “NgamsCfg.Log:LogBufferSize”). It should be defined such that it is not necessary constantly to open, write and close the log file, which costs time. Furthermore, it should not be so big that it is difficult to monitor the operation due to a long delay between an event and the time the log for the event is written in the log file. Note, that the log cache is always flushed when a command execution terminates.

It is possible to instruct the NG/AMS Server to log into the UNIX/Linux syslog. The level for logging into the syslog should be defined (CFG: “NgamsCfg.Log:SysLog”). Also a prefix, which is written in connection with

each log can be defined. This makes it possible to filter out log messages related to NG/AMS (CFG: “NgamsCfg.Log:SysLogPrefix”).

The NG/AMS Log File can be rotated with a given interval. The time given as an ISO8601 timestamp between each rotation must be specified (CFG: “NgamsCfg.Log:LogRotateInt”). It is also possible to specify the size of the history of rotated log files, i.e., how many rotated log files should be kept (CFG: “NgamsCfg.Log:LogRotateCache”).

1.18 Email Notification

Apart from the various types of logging described in section 1.17, it is also possible to instruct NG/AMS to notify various recipients about important events occurring via email.

The various types of Notification Message are:

Event	*Description*
Alert Notification	An Alert Message is generated as a result of a serious problem encountered. Such a problem may not be recoverable, and it is likely necessary to do some manual intervention. Normally preventative actions should be undertaken immediately.
Error Notification	An Error is the result of a problem encountered, which is not of a very severe character. Often an error situation is provoked by an external request, which is illegal for some reason. Could e.g. be that it is tried to archive a file when the system is in Offline State. Depending on the type of error, intervention should be undertaken (ASAP).
Disk Space Notification	A Disk Space Notification is sent out when a certain threshold of minimum free disk capacity is reached. This message is meant only as a 'warning' indicating that the Storage Set is about to be full. No actions are needed, apart from maybe verifying that Storage Sets with free disk space are available.
Disk Change Notification	A Disk Change Notification is send out to indicate that one or both disks of a Storage Set is full (completed) and should be removed from the archiving unit and normally replaced with a new Storage Set. See also section 1.19.
No Disk Space Notification	If no more free Storage Sets are available, a No Disk Space Notification Message is send out to the subscribers of this event. Since this is a severe problem for an archiving system, a special Notification Message is dedicated for referring to this specific problem.
Data Check Notification	If the Data Consistency Check Service encounters errors/problems with data files, a Data Error Notification Message is send to the subscribers of this event. The files that were found to be 'bad' in some way should be analyzed to find out what is causing the problem. It could be caused by physical problems of the disk, or that due to long storage on the disk, failures start to occur. Problems with a 'problematic' file are normally only reported once. I.e., if the problem is not solved, there will be no more notification about the problematic file until the NG/AMS Server is re-started.

Table 10: The different types of Notification Messages.

The Notification Setup is configured in the NG/AMS Configuration (CFG: "NgamsCfg.Notification"). For further information about this specific property see chapter 6. An example Disk Change Notification Message can be found in section 1.19.

In order to prevent flooding of Notification Emails, a mechanism is provided, which buffers messages of the same kind, once such a message has been sent out. Only when a certain number of messages or after a given time-out the subsequently messages are bundled in Notification Messages and send out. This is referred to as Email Notification Retention. See the description of the attributes for the Notification Element in the configuration (section 1.46). An example of an Notification Email containing several messages is shown in the following:

```
Subject: NGAS-ngahu2-7777: PROBLEM ARCHIVE HANDLING
Date:    Wed, 4 Feb 2004 16:16:44 +0100 (MET)
From:    ngas@eso.org

Notification Message:

ACCUMULATED NOTIFICATION MESSAGES:
```

```
--MESSAGE#000001/2004-02-04T14:44:48.209----------
NGAMS_ER_ARCHIVE_PUSH_REQ:4011:ERROR:  Problem  occurred  handling  Archive  Push  Request!  URI:  WFI.2001-02-
01T01:52:10.819.fits.Z.
--MESSAGE#000002/2004-02-04T14:58:29.303----------
NGAMS_ER_ARCHIVE_PUSH_REQ:4011:ERROR:  Problem  occurred  handling  Archive  Push  Request!  URI:  WFI.2001-02-
01T01:52:10.819.fits.Z.
--MESSAGE#000003/2004-02-04T15:01:30.073----------
NGAMS_ER_ARCHIVE_PUSH_REQ:4011:ERROR:  Problem  occurred  handling  Archive  Push  Request!  URI:  WFI.2001-02-
01T01:52:10.819.fits.Z.
--END-------------------------------------

Note: This is an automatically generated message
```

The Maximum Retention Size and the Retention Buffer Size, are defined by the parameters "NgamsCfg.Notification:MaxRetentionTime" and "NgamsCfg.Notification:MaxRetentionSize".

1.19 Disk Space Monitoring

During the archiving process, NG/AMS monitors constantly the state of the set of disks currently installed. If the amount of data on a Storage Set reaches a certain limit defined by a configuration parameter, a Notification Message can be sent out to a list of subscribers for this event (see 1.18). This event is a pre-warning that this Storage Set is going to be completed (full) within a limited time. The latter depends on the threshold defined in the configuration file. When a Storage Set is considered as 'completed', another type of Notification Message can be broadcast to a number of subscribers. This message will indicate that the Storage Set is full and needs to be replaced. The appearance of such an email message is as follows (example):

```
Subject: NGAS-w2p2nau-7777: CHANGE DISKS
Date:    Fri, 25 Jan 2002 01:06:26 +0100 (MET)
From:    ngasmgr.w2p2nau@eso.org

Notification Message:

PLEASE CHANGE DISKS:

Main Disk:
- Logical Name: FitsStorage-M-000024
- Slot ID:      5

Replication Disk:
- Logical Name: FitsStorage-R-000024
- Slot ID:      6
```

Figure 6: Disk Change Email Notification Message.

The Logical Name(s) (Disk Label(s)) as well as the Slot in which the disk(s) are hosted is indicated in the mail. When such a message is received by the NGAS responsible (operators) it is advisable to carry out the suggested changes as soon as possible to avoid saturation. If only a single disk in a set is completed, the Email Notification will only indicate the name of this completed disk (see the sections 1.12 and 1.14).

Also the disk space in the NG/AMS system storage areas, where log files, back-log buffered files etc are stored are continuously monitored. This is done by the Janitor Thread. If the amount of free disk space on one of the system files storage areas goes below a defined threshold (CFG: "NgamsCfg.JanitorThread:MinSpaceSysDirMb"), the NG/AMS Server in question will bring itself to Offline State and cannot handle further commands before the problem has been solved and the server brought Online.

1.20 Simulation Mode

It is possible to operate the NG/AMS Server in Simulation Mode, whereby a number of features are disabled or are working slightly different than in Normal Mode. One of the major differences is that it is possible to run without the availability of 'real' storage disks. Simulated storage disks are created as directories in the Mount Root Point. These are of the format: "<mount root point>/<storage set ID>-Main | Rep-<simulation slot ID>".

Another difference compared to running in 'real mode' is that the Online and Offline Plug-Ins are not executed since no disks need to be mounted or unmounted. The disk information about the disks is generated/simulated and written in the DB.

For the clients of NG/AMS there is no visible difference between running in Normal Mode or in Simulation Mode. Also the internal aspects are the same, so that e.g. the DB is updated in the same manner in Simulation Mode as in Normal Mode. The Simulation Mode can be quite useful for developing and testing e.g. DAPIs and DPPIs.

It is possible to have a fully operational NG/AMS installation running in Simulation Mode on a 'normal' workstation archiving and retrieving data to/from one of the system disks of the workstation.

To enable/disable the Simulation Mode, the attribute "NgamsCfg.Server:Simulation" in the configuration is used. See also chapter 6.

Another way to obtain simulation is to implement an Online Plug-In, which registers the volumes found under the Volume Directory (see 1.12). This could be obtained by creating ‘volume directories’, which are identified by a certain file, located in the volume root point, e.g. “<Volume Root Directory>/.ngas_volume_id”. Latter, is also useful for managing RAID volumes.

1.21 Back-Log Buffering

Back-Log Buffering is used to temporarily buffer data, which for some reason, not necessarily related to the quality of the data, prevents NG/AMS from performing a proper archiving of the data file. An example of such an event, is e.g. if the DB connection is lost temporarily.

As shown in ***Figure 3***, the Back-Log Buffer Area could be located in the NG/AMS Root Mount Directory as it is practical to collect the data of NG/AMS under a single point. If a problem occurs during the handling of an Archive Request, a file with a unique name will be created in this area and the data of the request buffered in this file. The reply to the Archive Request will indicate the problem, i.e. that the data was Back-Log Buffered. No further actions are needed from the client that issued the Archive Request. ***Figure 7*** shows an example of a reply from NG/AMS when back-Log Buffering was done.

```
ngasmgr@acngast1:/opsw/NGAS/ngams/ngamsData> ngamsCClient -port 7777 -host acngast1 -status -cmd ARCHIVE -fileUri
~/tmp/SmallFile.fits

<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
  <Status Date="2003-01-08T16:44:40.562" HostId="acngast1" Message="NGAMS_WA_BUF_DATA:4015:WARNING: Problems
          occurred while handling file with URI: SmallFile.fits. Data will be  buffered, and attempted archived
          at a later stage. Previous error  stack: NGAMS_ER_DB_COM:2002:ERROR: Problems communicating with the
          DB: Error: connection is not open." State="ONLINE" Status="FAILURE" SubState="IDLE"
          Version="v2.0-Beta2/2002-12-04T09:22:53"/>
</NgamsStatus>
```

Figure 7: Example reply when Back-Log Buffering is applied.

The NG/AMS Server has an internal thread, Janitor Thread, which runs periodically and tries to clean up the NG/AMS environment. One of the tasks performed is to archive Back-Log Buffered data. If such an attempt fails due to one of the reasons justifying for Back-Log Buffering, the data will be kept in the Back-Log Buffer and a new attempt to archive it repeated later. If the attempt fails for another reason, the data will be moved to the Global Bad Files Area shown in ***Figure 3***. In this case a Notification Message will be sent out to the subscribers of Error Notification Messages, and the appropriate information logged in the log output targets specified (see the sections 1.17 and 1.18).

In the NG/AMS Configuration it can be specified if Back-Log Buffering should be performed, as well as the parent directory for the Back-Log Buffer Directory. For further information about this specific property consult chapter 6.

1.22 The NG/AMS Server Command Interface

The NG/AMS Server command interface is based on the standard HTTP protocol. This makes it possible to interface to the NG/AMS Server from different kinds of clients in a simple and straightforward manner. E.g. from a WEB browser (better if XML enabled) it is possible to query the status of an NG/AMS Server:

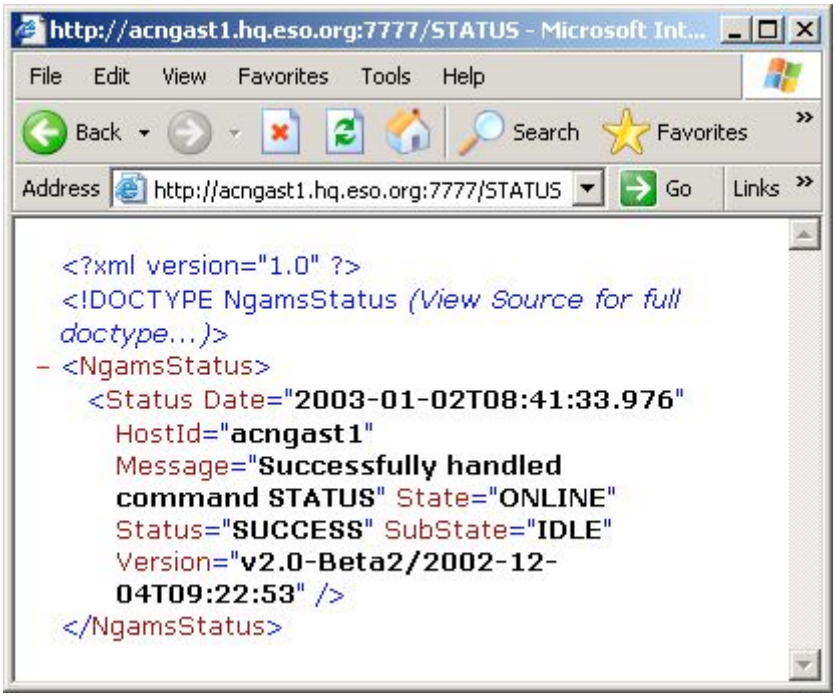

Figure 8: Interaction with an NG/AMS Server from a WEB browser.

Also a standard utility like "telnet" can be used to interact with NG/AMS, e.g. to issue a command like OFFLINE[5]:

```
ngasmgr@acngast1:/opsw/NGAS/ngams> telnet acngast1 7777
Trying 134.171.21.30...
Connected to acngast1.
Escape character is '^]'.
GET STATUS

HTTP/1.0 200 OK

<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
  <Status Date="2002-12-23T14:59:42.724" HostId="acngast1" Message="Successfully handled command STATUS"
          State="ONLINE" Status="SUCCESS" SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
</NgamsStatus>
Connection closed by foreign host.
ngasmgr@acngast1:/opsw/NGAS/ngams>
```

Figure 9: Example of interaction with NG/AMS using "telnet".

In general, the NG/AMS Python and C based command interface tools, should be used when interacting with NG/AMS from the shell. See section 1.41 for more information about these tools.

An upper limit for the maximum amount of requests that can be handled in parallel can be defined in the NG/AMS Configuration (CFG: "NgamsCfg.Server:MaxSimReqs"). If it is attempted to issue a request which results in a number of requests larger than the one defined in the configuration, such new requests are rejected by the system and will have to be issued again at a later stage.

For more in-depth information about the NG/AMS command interface, consult the chapters 7 and 16.

The NG/AMS Server maintains an internal DB with information about requests being handled and which were handled. It is possible to query the status of such a request via the command interface, by issuing the command "STATUS?request_id=<request ID>". In this way, the status of e.g. a Clone or Register Request can be supervised.

[5] This is possible, only if HTTP Authorization is disabled (section 1.26).

1.23 Data Consistency Checking

The NG/AMS Server can be configured to carry out a periodic consistency check of the data files, which are stored on the disks installed on that NGAS Node. The following checks are carried out:

- It is checked if files are registered in the DB but are not found on the disk.
- Checksum value for each file is checked according to the value registered in the NGAS DB for the file.
- It is checked if files are found in the Storage Area of the storage disks, which are not registered in the DB.

In case discrepancies are found in the data holding on the disks in connection with an NGAS Host, a Data Inconsistency Notification Message is send out. This has the format, e.g.:

```
Subject: NGAS-arcus2-7778: DATA INCONSISTENCY(IES) FOUND
Date:    Fri, 25 Jan 2002 01:06:26 +0100 (MET)
From:    jknudstr@eso.org

Error Message:

DATA INCONSISTENY(IES) FOUND IN DATA HOLDING:
Date:            2002-02-12T15:32:05.424
NGAS Host:       arcus2
Inconsistencies: 1

Problem Description                    File ID                          Version
-------------------------------------------------------------------------------
ERROR: Inconsistent checksum found     TEST.2001-05-08T15:25:00.123     3
-------------------------------------------------------------------------------
```

Figure 10: Example of a Data Consistency Checking Report.

If files are found, which do not have the checksum properly set, NG/AMS will calculate the checksum using the DCPI specified in the configuration, and send a Data Inconsistency Notification Message to the subscribers of this type of message.

It is possible to enable and disable the Data Consistency Checking Service (CFG: "NgamsCfg.DataCheckThread:DataCheckActive"). In addition it is possible to allocate a priority to the data checking thread to calibrate the CPU consumption (CFG: "NgamsCfg. DataCheckThread:DataCheckPrio"). It is also possible to specify how disks and files are checked, whereby this can either be done sequentially or randomly (CFG: "NgamsCfg.DataCheckThread:DataCheckDiskSeq", "NgamsCfg.DataCheckThread:DataCheckFileSeq"). A minimum cycle time for one iteration of the service can also be defined (CFG: "NgamsCfg.DataCheckThread:DataCheckMinCycle"). If the checking is carried out in less than the specified minimum cycle time, the service will be suspended for a while. A parameter is used to configure the service to produce a log entry after each iteration with summary information about the check carried out. This log entry has the following contents (example log entry taken from the Local Log File):

```
2002-02-26T02:52:00.640 [INFO] Number of files checked: 9529. Amount of data checked: 582478.078 MB. Time for
checking: 25139.280 s
```

Figure 11: Example of a Data Consistency Checking Status Log.

The Data Consistency Checking spawns one Checking Thread per disk registered in an NGAS Node. This is done to speed up the process of checking the data since it gives a better utilization of CPU and disk I/O. In particular in a system with several CPUs this is an advantage. The amount of such threads should be approximately twice the amount of CPUs in the system. The maximum number of such threads, is defined in the configuration (CFG: “NgamsCfg.DataCheckThread:MaxProcs”.

Normally when executing a Data Consistency Check, the of the files is computed and checked against the checksum registered for that file in the NGAS DB. The type of checksum (the Checksum Plug-In) is given in the DB. If it is desirable to perform only a scan of the data holding, the system can be configured to carry out the checking in this way (CFG: “NgamsCfg.DataCheckThread:Scan”). If a Data Consistency Check Scan is requested, it is only tried to access the files and the file size is checked against the size registered in the NGAS DB. This is of course much faster than calculating the checksum.

Usually only when errors in the data holding are found, a Data Check Notification Message is sent out to the subscribers. It is possible however to force the system to send out a Notification Message each time a Data Consistency Check cycle has been executed (CFG: “NgamsCfg.DataCheckThread:ForceNotif”).

The configuration parameters mentioned above are described in more detail in chapter 6.

1.24 Label Printing

A label to stick on the disk cases can be produced by NG/AMS by means of the LABEL Command. The text on the label is the Logical Name allocated to the disk. In addition the host ID and the Slot ID are printed on the label. An example of a label is as follows (generated by the Brother P-Touch 9200 DX label printer):

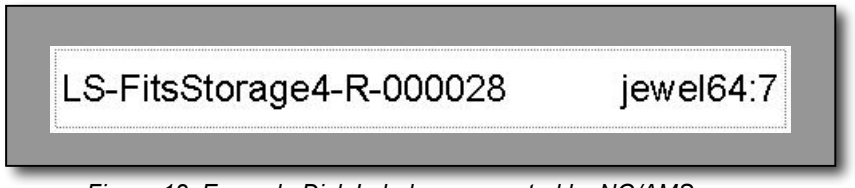

Figure 12: Example Disk Label as generated by NG/AMS.

The part with the Host ID + Slot ID should be removed from the label before sticking it on to the disk case.

The LABEL Command takes as input the Disk ID for the disk to be printed. The label is printed by the Label Printer Plug-In (see chapter 1.60).

It is also possible to use the LABEL Command to re-label (rename) disks. The new label must then be given and this is changed in the NGAS DB and in the NGAS Disk Info file on the disk. Subsequently a new label can be printed, again using the LABEL Command.

1.25 Service Privileges

NG/AMS provides a simple scheme for defining which overall actions can be executed by an NG/AMS Server. These high level NG/AMS services that can be enabled/disabled via the NG/AMS Configuration are:

Service	*Description*
Archive Request Handling	It is possible to enable/disable acceptance of Archive Requests. This is e.g. relevant in an archive data server cluster configuration where no files are being archived. Note that if archiving is disabled, apart from the ARCHIVE Command also the CLONE Command will not be accepted. In addition it will not be possible for NG/AMS to handled back-logged buffered data nor will it be possible to act as Data Subscriber (CFG: “NgamsCfg.Server.AllowArchiveReq”).
Retrieve Request Handling	Disabling of handling of Retrieve Requests, may be applied for NGAS Node used as archiving units, where it is not desirable that handling of external data retrieval disturbs/loads the system (CFG: “NgamsCfg.Server:AllowRetrieveReq”).
Processing Request Handling	In connection with a Retrieve Request it is possible to specify that data processing should be applied on the data before replying to the requestor. This may be relevant to avoid to load an NGAS Host too much if handling of the Retrieve Requests themselves is high-priority and where processing would load the system too much to get access to the data within a limited period of time (CFG: “NgamsCfg.Server.AllowProcessingReq”).
Remove Request Handling	If this feature is disabled, no REMFILE, REMDISK and DISCARD Commands will be accepted by the NGAS Host, and it is thus not possible to delete any information in the NGAS system. This should usually be applied e.g. for NGAS Nodes operating in an NGAS data server cluster (CFG: “NgamsCfg.Server.AllowRemoveReq”).

Table 11: NG/AMS High Level Services that can be enabled/disabled.

See also section 1.46/”Server” Element.

Apart from disabling handling of REMFILE/REMDISK/DISCARD Commands (***Table 11***), it might be advisable to implement additional schemes for preventing data from being deleted from an archive data server NGAS system. This could e.g. be done within the System Online Plug-In (see chapter 1.58). One of the responsibilities of this plug-in is to mount the Storage Media available in the NGAS Node. In case a Storage Media is marked as ‘completed’ in the NGAS DB, the media could be mounted read-only to prevent data from being (accidentally) removed. NG/AMS does not provide any features to prevent data from being as such, and it is up to the designers of the NGAS environment to define how to provide this.

1.26 Access Restriction

In order to prevent unauthorized access to NGAS Nodes, NG/AMS implements a scheme for authorization. This is based on HTTP authentication (basic scheme). Using this scheme a user name and password must be provided (encrypted). A simple encryption scheme is used[6]. This is also supported by most WEB browsers.

The users, which are authorized to connect to an NGAS system, must be defined in the NG/AMS Configuration (CFG: "NgamsCfg.Authorization"). For each user a password must be defined and must be given encrypted in the definition.

In the future the authorization will be extended such that it will be possible to define per user, which services are allowed.

1.27 Janitor Thread

The Janitor Thread is a process which runs within the NG/AMS Server as a background task. Its purpose is to keep the operational environment of the NG/AMS clean and tidy and to carry out a number of different tasks, which is related to the system.

The complete list of tasks maintained by the Janitor Thread is as follows:

Task	*Description*
Update the DB Snapshot DB (DBM) for each disk during initialization	When the server starts up, the Janitor Thread checks if the DB Snapshot (section 1.52) on each disk is in sync with what is registered in the DB. If not, the DB and or DB Snapshot is/are updated accordingly. Note, if there are many disks to be check with many files on them, this procedure may take a while.
Update DB Snapshot when new files are archived or deleted	During operation, whenever changes are introduced in the NGAS DB in connection with files, this is updated 'real-time' in the DB Snapshot. The DB Snapshot should therefore normally be up-to-date when the server goes Offline.
Archive Back-Log Buffered files if such are found in the Back-Log Buffer	If files could not be archived due to e.g. DB connection problems (section 1.21) and thus were Back-Log Buffered, these will later be attempted archived by the Janitor Thread If the server is not successful attempting this, it will re-try periodically if the problem encountered qualifies for keeping the file in the Back-Log Buffer. Otherwise, the file is removed to the Bad Files Directory.
Clean up Processing, Temporary Files Directory	The Janitor Thread continuously monitors the contents of the Temporary and Processing Directories defined for the NG/AMS Server. If files are found, which are expired, i.e., which have not been used since a give time-out, these files are removed to keep the system tidy.
Clean up Request Queue	The NG/AMS Server maintains a internal DB with information about the requests handled. The Janitor Thread periodically cleans up entries, which have expired from this queue.
Clean up Subscription Back-Log Buffer if files in this buffer have expired	The NG/AMS Subscription Service may Back-Log Buffer files, which could not be delivered to a Subscriber. If these files remain for a longer period of time in the Subscription Back-Log Buffer, these are removed by the Janitor Thread to avoid saturation of the system disks. The expiration time for such files are defined in the configuration (CFG: "NgamsCfg.SubscriptionDef:BackLogExpTime").
Send out Retained Notification Email Messages if the condition is met	If email messages are found in the Email Notification Retention Buffer, which are expired, the Janitor Thread purges the buffer for these messages, i.e., they are sent out.
Rotation of NG/AMS Log File	The Janitor Thread will rotate the NG/AMS Log File with a periodic interval defined in the configuration. The rotated log file will remain for a while in the directory hosting log files.
Cleaning up rotated log files	If there are more rotated log files buffered than specified in the configuration, the oldest log files are deleted such that the number of log files matches the defined size of the history.
Check if there is enough disk space available on	The Janitor Thread continuously monitors the amount of free space on

[6]RFC 1521 (MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, section 5.2, "Base64 Content-Transfer-Encoding").

the various system disk areas used by the NG/AMS Server	the various system disks used during operation of the server. If the amount of free disk space of one of these areas goes below the limit defined in the configuration, the Janitor Thread forces the server to go Offline, such that no further services are available. Only EXIT, STATUS and ONLINE Commands can be issued. A server in this state refuses to go Online if the amount of disk space is not adequate.
Check if other nodes have requested to be woken up by this node	If other nodes in e.g. a cluster have suspended themselves, they may request to be woken up. If such a wake-up call is placed to an NGAS Node on which the NG/AMS Server is running, the Janitor Thread will wake up the node in question if the time for waking up the suspended node is passed.
Check if the node should be suspended	The Janitor Thread checks periodically if the node on which it runs should be suspended. If this is the case, the Janitor Thread invokes the Suspension Plug-In defined (section 1.30).

Table 12: Tasks maintained by the NG/AMS Janitor Thread.

The cycle time for the Janitor Thread must be defined in the NG/AMS Configuration (CFG: “NgamsCfg.JanitorThread:SuspensionTime”).

4. *EXPERT:* Advanced Features

In this chapter the advanced features of NG/AMS listed in section 1.9 are described in detail.

1.28 EXPERT: Operation in Cluster Mode

For larger data holdings, it will normally be necessary to have a maybe large number of NGAS Nodes to provide online access to the data in the Archive Facility. It is therefore important to make a proper design of the architecture of such a Archive Facility Cluster.

NG/AMS provides a few services to support such operation in 'cluster mode'. Among this is the capability of NG/AMS to act a proxy while handling a Retrieve Request (see also section 1.16). In addition NG/AMS distinguishes between NGAS Nodes globally accessible, and nodes within a 'cluster network' (private network). With this simple scheme, it is possible to build up e.g. a hierarchical cluster as the one shown in ***Figure 13***.

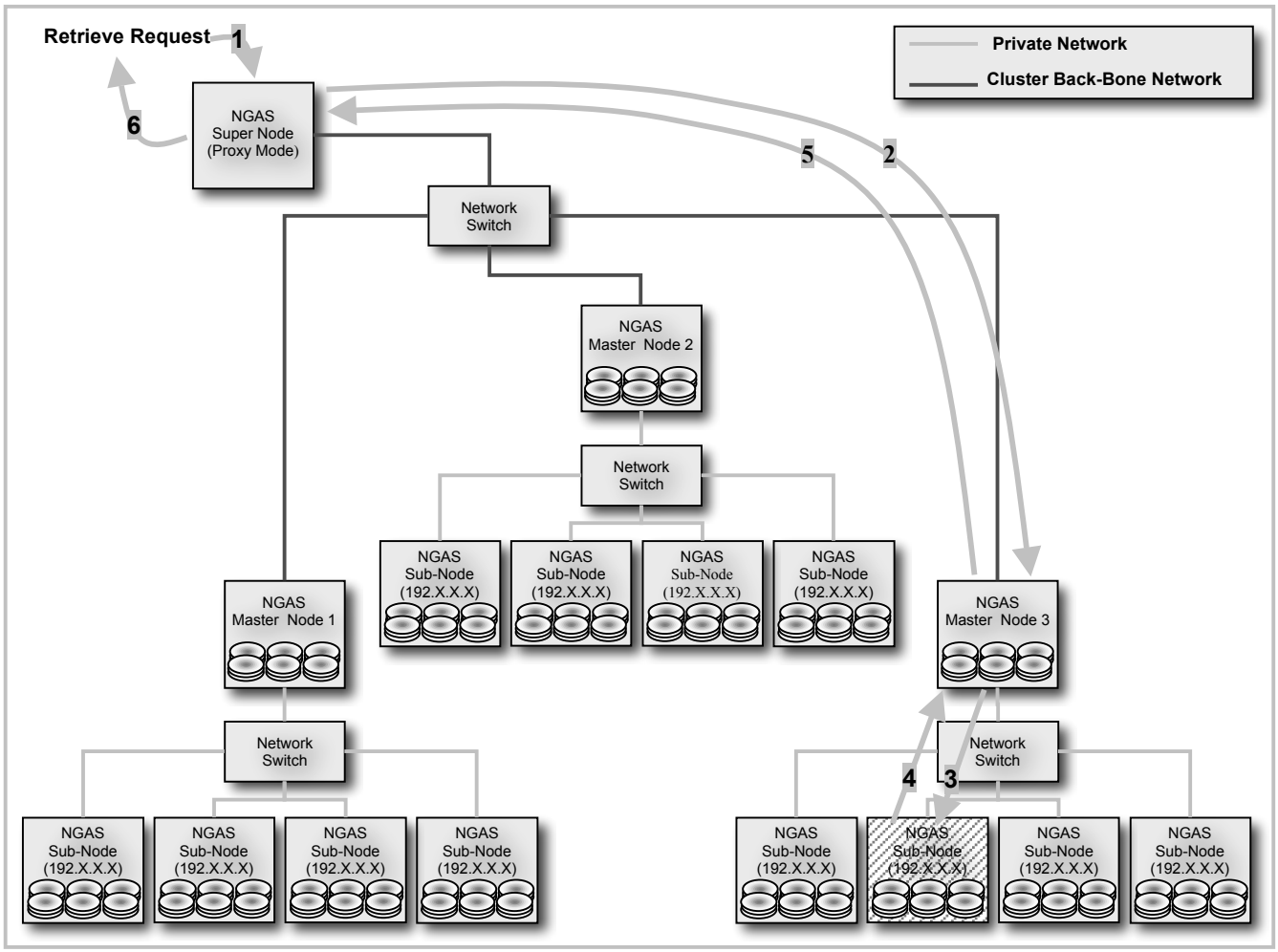

Figure 13: Example of an NGAS Cluster.

In the cluster shown in Figure 13, the main entry point of the NGAS Cluster (and the only one for that matter), is the NGAS Super Node. All requests must pass through this node. When a Retrieve Request is received by the super node (1), it will identify that the file requested is located on the NGAS Node high-lighted in the figure. This it finds out from the "ngas_files", "ngas_disks" and "ngas_hosts" tables in the NGAS DB (DB: "ngas_files.-disk_id" → "ngas_disks.host_id" → "ngas_hosts.host_id"). From the Cluster Name defined for the node, it determines if it can connect directly to the node or if it should go via the Master Node. From the "ngas_hosts" table it finds the NGAS Master Node for that sub-cluster (NGAS Master Node = "ngas_hosts.cluster_name"), and it forwards the Retrieve Request to the NGAS Main Node 3 (2). The NG/AMS Server on NGAS Master Node 3 in turn figures out that the file is located on a disk hosted in a node within 'its cluster network. It therefore

retrieves the file via the private network (3 and 4). If processing was requested this is carried out on the sub-node. Subsequently the NGAS Main Node 3 sends back the final result to the NGAS Super Node (5). The NGAS Super Node in turn, returns the result of the Retrieve Request to the external requestor (6).

The scenario in ***Figure 13***, shows a rather complex environment. The example serves mainly to illustrate the capabilities of the NG/AMS SW. It is probably always an advantage to have one single entry point to an NGAS Archive to make it easy for external clients to access the data. In addition, for security reasons it is an advantage to have only one such entry point to the archive cluster. A disadvantage of this scenario is that each request for data will have to pass through two NG/AMS Servers acting as proxies before arriving to the client. This of course means an extra overhead.

If the Suspension/Wake-Up Service is used (see section 1.30), it is important that each suspended host, is accessible by one other NGAS Host, which is never suspended and therefore can be requested to wake up such a suspended host. In the example in Figure 13, the sub-nodes could be suspended, whereas the main nodes will have to be kept running with an NG/AMS Server in Online State running on them.

Another and simpler example of an NGAS Cluster is shown in ***Figure 14***.

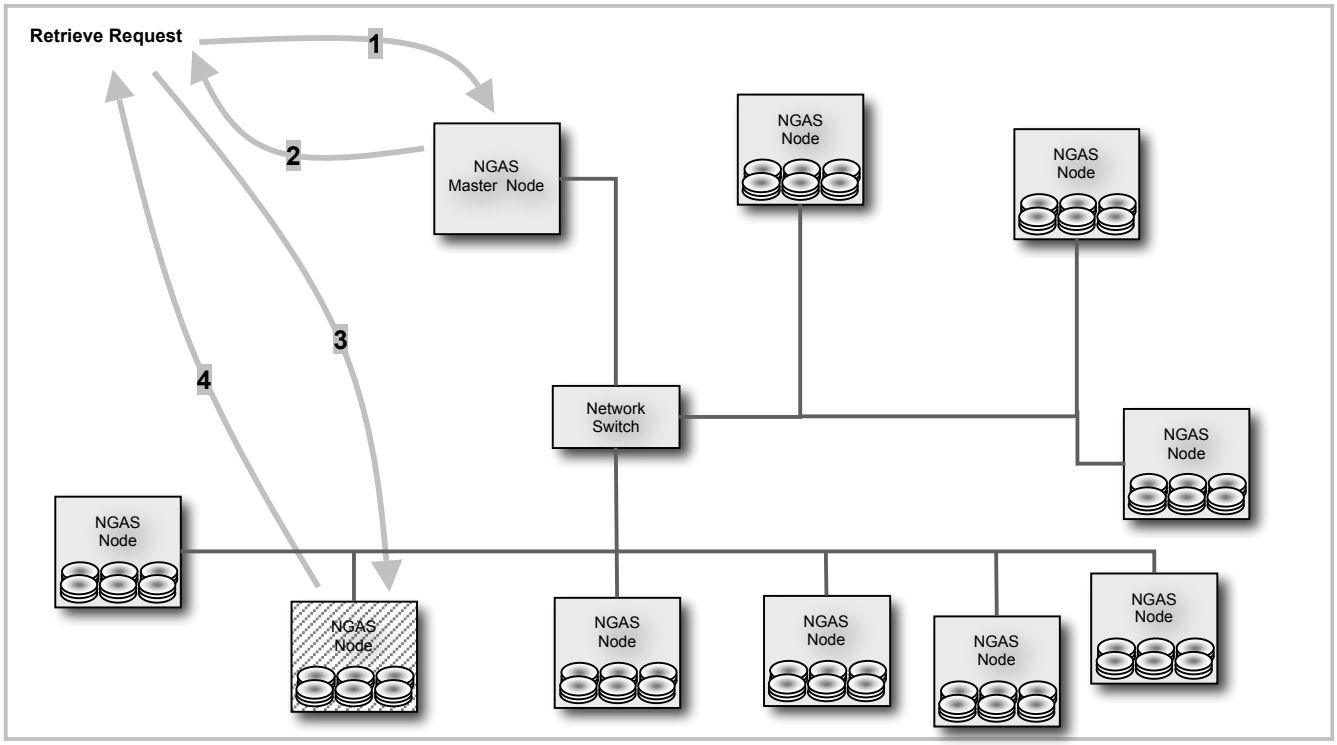

Figure 14: Example of a 'simple' NGAS Cluster.

The architecture in ***Figure 14*** is based on a 'flat structure' providing *external* access to the individual node in the cluster. The client has still one contact point as in the scenario in ***Figure 13***, namely the NGAS Master Node and it sends all Retrieve Requests to this node (1). In this case however, the NGAS Master Node does not act as proxy, and after identifying on which NGAS Node the file is located, it returns an HTTP Redirection Response to the requestor (2). The client now issues the same Retrieve Request directly to the NGAS Node where the file is located (3). The NG/AMS Server on that host handles the request and possibly processes the file and sends this back, *directly* to the requestor (4).

It goes without saying, that the structure shown in ***Figure 14*** makes the handling of requests far more efficient compared to the structure used in ***Figure 13***. There is however still the issue of security to take into account. I.e. all nodes are accessible externally. In addition, the file access is no-longer transparent, since the client has to support the re-direct scheme defined by the HTTP protocol. Using the C or Python-APIs or command utilities however, this is handled transparently for the client (see the chapters 9 and 10, and section 1.40).

As a last example scenario an architecture similar to the one in ***Figure 14*** could be used, whereby there is one NGAS Master Node acting as main entry point. All nodes however, in the cluster are connected to the switch via a private/cluster network, which is not accessible externally. This topology has the advantage of still having only one contact point for external clients. At the same time access to the data is handled transparently, as the NGAS Main Node will always act as proxy on behalf of the client. There is only one intermediate copy created (due to the proxy mode) using this architecture (as opposed to two proxy copies in the scenario in ***Figure 13***). The processing is carried out on the individual NGAS Nodes.

Other architectures may be designed using the NG/AMS Server and NGAS Nodes in various configurations. The three architectures discussed in this section, merely serve to give an impression of what capabilities are provided by the NG/AMS SW.

1.29 EXPERT: Data Subscription Service

The Data Subscription Service of NG/AMS, makes it possible to synchronize data holdings of different NGAS Nodes partially or completely and to export data on-the-fly from one NGAS archive to remote sites that need part of the data or all data becoming available on an NGAS Host. A client subscribing for data is referred to as a Data Subscriber. An NG/AMS Server, which delivers data to such a Subscriber, is referred to as a Data Provider.

When a client subscribes, it can specify to receive data from a certain point in time. In this way it is possible for a client to receive older files. Otherwise, the time for subscription is taken as starting point and only new files archived from the time of the subscription are taken into account for that client.

It is also possible to specify a Filter Plug-In (see chapter 1.67), which is applied on the data files to determine whether or not to deliver the file to a specific Data Subscriber. A client subscribes itself by issuing a SUBSCRIBE Command (see also section 1.97).

The client subscribes itself giving a so-called Subscriber URL to the Data Provider NG/AMS. NG/AMS delivers data to the client by performing an HTTP POST on the Subscriber URL. It is up to the client to specify a proper Subscriber URL. On the client side a corresponding HTTP server must be ready to handle the data delivery POST requests from the Data Provider. Any WEB server can be used, from a simple customized implementation to an existing and widely used server like e.g. Apache[7]. The server must of course be capable of handling the data delivery request. The handling could be implemented as a CGI script.

An NG/AMS Server can be configured to subscribe itself as a Data Subscriber to another NG/AMS Server. In this case the Subscriber URL should be the URL used when performing an Archive Push Request, i.e. "http://<host>:<port>/ARCHIVE" and the corresponding DAPI should be made available within the context of the Data Subscriber NG/AMS Server to handle the possible types of data that can be delivered. In order to make an NG/AMS Server subscribe itself, the configuration must be adjusted accordingly (CFG: "NgamsCfg.SubscriptionDef", see chapter 6). It is possible to instruct an NG/AMS Server to un-subscribe itself automatically when it goes Offline (CFG: "NgamsCfg.SubscriptionDef:AutoUnsubscribe").

When a client subscribes, it can allocate a priority to itself. This priority determines how much CPU time the delivery of data files to that client may consume. A client that subscribes itself with a lower priority than other Subscribers will receive the files later than these other Subscribers. It should be evaluated carefully for each client how soon the data should be delivered. The default priority is 10. The lower the priority number, the more CPU time the client is allocated. I.e., "0" is the highest available priority. It is advisable to allocate such a priority with great care since the data delivery might consume a lot of CPU time, and may interfere with more urgent Archive or Retrieve Requests.

When a client has first subscribed itself to a certain type of data, NG/AMS guarantees that all files of that type and matching the time constraint, will be delivered to the client. If it is impossible to deliver a file if e.g. the client has terminated execution or due to interruption of the network connection, NG/AMS will back-log buffer the data in the Subscription Back-Log and try periodically to deliver the data to the client. Even if the Storage Media hosting the files to deliver to the client are removed from the Data Provider NGAS Host, the files will be

[7] http://www.apache.org.

delivered, since the Subscription Back-Log is located in a separate area. Note that normally the Back-Log Area should be located on one of the permanent disks of the NGAS Host to facilitate this scheme.

Beware that if files cannot be delivered during a longer period of time, the back-log storage area may fill up with Back-Log Buffered files.

It is possible to specify an expiration period of time indicating for how long time data should be kept in the Subscription Back-Log (CFG: “NgamsCfg.SubscriptionDef.BackLogExpTime”). Data residing longer than the expiration time will be deleted and thus never delivered. The name of the Subscription Back-Log is defined by the parameter: “<NgamsCfg.Server:BackLogBufferDirectory>/subscr-back-log”. A table in the NGAS DB is used to keep track of this Subscription Back-Log (section 1.51).

A simple scheme has been implemented to avoid that the same data file is delivered several times to a client. This scheme is based on recording the ingestion date for the last file delivered. I.e., only files with a more recent ingestion date will be taken into account. This remembered ‘last ingestion date’ for each client will be reset if a start date for the subscription ‘older’ than this date is specified by a client.

A Data Subscriber unsubscribes itself by sending an UNSUBSCRIBE Command. The client remains subscribed as soon as it has sent the SUBSCRIBE Command until an UNSUBSCRIBE is submitted. When a client issues the UNSUBSCRIBE Command, the Subscription Back-Log for that client will be reset and thus possible Back-Log Buffered data will not be delivered.

Care should be taken to avoid ‘circular subscriptions’, i.e., that two clients subscribe to each other for the same type of data. In such a case, the two serves would continue to deliver the file to each other, ending up saturating the system. A Subscriber cannot subscribe to itself.

It is possible to switch off the Subscription Service globally via the configuration (CFG: “NgamsCfg.SubscriptionDef:-Enable”). The subscription service is handled by an internal thread (Data Subscription Thread) running within the NG/AMS Server. It is possible to specify how often this thread should be scheduled in the configuration (CFG: “NgamsCfg.SubscriptionDef:SuspensionTime”). This suspension time determines how often the server will try to deliver Subscription Back-Log data. The Data Subscription Thread is scheduled explicitly when new data become available on an NGAS Host. The suspension time, defines how frequently the thread should try to deliver Subscription Back-Log Buffered data.

1.30 EXPERT: Server Suspension/Wake-Up Service

Since an NGAS Host may be idling for longer period of times, it is relevant to suspend such a host. This is relevant, in particular in case of clusters of NGAS nodes, which consume a non-negligible amount of power. A feature is provided by NG/AMS whereby it is possible to configure an NG/AMS host to suspend itself after a certain period of idle time (CFG: “NgamsCfg.HostSuspension:IdleSuspensionTime”). Host suspension can be enabled/disabled globally via the configuration (CFG: “NgamsCfg.HostSuspension:IdleSuspension”).

When an NG/AMS Server identifies that it should suspend itself, it invokes the so-called Suspension Plug-In (see chapter 1.65), which actually takes care of suspending the system. Apart from various/possible clean-up of the system, this usually simply means to shut down the NGAS Host. The host should normally be configured such that when a shut-down is performed, the NG/AMS Server is terminated in a clean manner.

After suspending itself, an NGAS Host can only be ‘woken up’ by ‘external intervention’. This means that either the host must be switched on manually, or the server must request to receive a ‘wake-up call’ from another NGAS Host. An NGAS Host suspending itself, signals by which other NGAS Host it would like to be woken up (CFG: “NgamsCfg.HostSuspension:WakeUpServerHost”). This means that in an NGAS Cluster where host suspension is used, one host should be kept switched on with an NG/AMS Server running in Online State. An NG/AMS Server suspending itself, will calculate when it should be woken up at latest. This is determined by the time for scheduling the next data checking batch if Data Consistency Checking is active.

A suspended server will also be woken up, if a request for data located on the suspended host is received. In order for this to work, all Retrieve Requests must pass through one node in the NGAS Cluster (main node), which should never be suspended. The main node will identify that the requested data is stored on a suspended host, and will wake up this node as described above. Handling a Retrieve Request of data stored on a suspended host, may therefore take some time depending on how long time it takes the host to

	become operational (Online). A proper time-out must therefore be applied when retrieving data from an NGAS Cluster where host suspension is used. Once the suspended server is Online, requests will be handled rapidly, until it is suspended again (after the specified period of idle time).

An NG/AMS Server, which is requested to wake-up a suspended NGAS Host, will invoke a Wake-Up Plug-In when the time for waking up the suspended host has arrived (CFG: "NgamsCfg.HostSuspension:WakeUpPlugIn"); see also chapter 1.66 for more information about the Wake-Up Plug-In. The Wake-Up Plug-In will usually inform some device connected to the network to switch on the suspended NGAS Host. After having launched the Wake-Up Plug-In, the NG/AMS Server will wait for the suspended NGAS Host to become active. If this does not occur within a certain time-out (CFG: "NgamsCfg.HostSuspension:WakeUpCallTimeOut") an error message is logged and an Error Email Notification Message send to the subscribers of this.

1.31 EXPERT: User Server Commands

It is possible to add new commands to the existing set of commands provided by the NG/AMS Server (see 16). These commands are referred to as "User Server Commands".

In order to add a new command to the server command interface, a plug-in must be provided. The file hosting the plug-in must have a name of the form "ngamsCmd_<Command>.py" and it must be installed in a directory, searched by the Python loader. It is not necessary to change the configuration to add new commands.

For an exact description of the User Command Plug-In, consult section 1.69.

1.32 EXPERT: Mirroring Service

The NGAS Mirroring Service is used to mirror the contents of one NGAS Archive with one or more other NGAS Archives. The mirroring can be set up to be bi-directional.

An NGAS Archive is here considered as a name space, defined by the NGAS Cluster definition, in the NGAS Hosts Table ("ngas_hosts.cluster_name") in the NGAS DB. This means the mirroring service uses the cluster concept when deciding which data object to mirror between the specified NGAS Clusters.

The mirroring service can be used to mirror data quasi on-the-fly (with a certain delay though) or to restore data, lost in one NGAS Cluster from duplicate data stored in other NGAS Sites.

One of the fundamental concepts of the NGAS Mirroring Service is that while designing the service, it has been attempted to spread out the load of carrying out the mirroring, on as many nodes in the target cluster as possible, to increase the throughput. Noticeable, not data is flowing through the instance controlling the mirroring, but all data is being mirroring from source to target node, for the maximal throughput.

1.33 EXPERT: Basic Functioning

An NGAS Cluster Name Space is defined by a logical view on the data holding in the NGAS System, which is contained within nodes, pertaining to one, specific cluster. The handling of the Cluster Name Space is managed at DB query level. By always considering nodes belonging to one cluster, according to the definition in the NGAS DB, it is possible to delimit the data objects belonging to each NGAS Cluster. This is utilized when determining whether to mirror data objects from one cluster to another.

Importantly, it is possible to set up the mirroring between NGAS Sites using NGAS DBs hosted in different RDBMS'. This makes it also possible to mirror data between an NGAS Sites using Oracle as the core RDBMS to another NGAS Site using Sybase as RDBMS.

Within each NGAS Cluster, one node should be assigned to become the Mirroring Control Unit, MCU. This unit takes care of coordinating the mirroring of data from a remote NGAS Cluster to the local one. As no data is passing through the MCU, this is not loaded with unnecessary data traffic, and will be merely controlling the data flow between the various storage units in the source and target cluster.

If mirroring should be carried out in the other direction, an MCU should be assigned also in the remote NGAS Clusters with which the local NGAS Cluster is synchronizing itself.

It is possible for an NGAS Cluster to synchronize itself with as many remote NGAS Clusters as required, whereby synchronizing with many sites, hosting larger amounts of data objects, may end up loading the target MCU significantly.

When activated, the MCU will run an internal thread, Mirroring Control Thread, which will take care of administering the mirroring from the remote sites to the local one. This thread will make use of a number of helper threads, which take care of checking for the availability of candidate data to be mirrored in the local cluster, and in case of missing data objects, re-archiving of this data from the source node to a node with storage capability in the local cluster.

Each data object selected for mirroring from the source to the target cluster, is referenced as a Mirroring Request, MR.

A table in the NGAS DB, NGAS Mirroring Queue Table, is used to manage the mirroring. This table can be used by external applications to provide the operators a status overview of the mirroring service activities.

1.34 EXPERT: Mirroring Schemes

The mirroring can be set up to apply the following schemes:

Selection Criteria	*Description*
No selection criteria	All data in the source cluster is considered as candidate data for mirroring to the local cluster. This can be used to make a complete synchronization of the local cluster with the remote, source clusters. Beware this check may be heavy to carry out, such that this should not be done too frequently, maybe only once or twice per 24 hour. Applying this, data lost in the local cluster can be restored from the remote data holdings.
Ingestion Date	Only files with an ingestion date, larger than the specified lower limit, will be considered for mirroring from the remote cluster. This is used to carry out a quasi on-the-fly, partial mirroring. This is lighter compared to the previous scheme as only new data is considered, in a window moving forward in time.
Mirroring Filter Plug-In	Filter Plug-In will be invoked on the remote NGAS Cluster to determine if data objects should be mirrored or not.
Ingestion Data and Mirroring Filter Plug-In	This scheme is a combination of the previous two, whereby only data objects, ingested after a given ingestion date are considered. On this data set, a specified Filter Plug-In is applied to finally determine whether or not to mirror the data.

For a normal set-up, the Ingestion Date Mirroring Scheme will usually be executed on a frequent basis, say, every 5 minutes to synchronize new data that has arrived in the source cluster combined with an entire synchronization a few times per day.

1.35 EXPERT: Configuration

The Mirroring Service is defined in the NG/AMS Configuration. The XML Element is named Mirroring. The attributes and sub-elements are defined as follows:

Attribute	*Description*
Mirroring.Active	Used to switch on/off the mirroring globally for that NGAS Server (NGAS Mirroring Control Unit).
Mirroring.Threads	The number of threads to run in the NGAS Server to actually handle the scheduled MR's. These threads are controlled by the Mirroring Control Thread (created, started, stopped, and paused).
Mirroring.CleanUpTime	Indicates a time (or list of times) when the Mirroring DB Queue should be flushed, i.e. handled entries removed.
Mirroring.Report	Gives a list of Email Notification Recipients, which will receive a periodic report about the activities of the mirroring. If not specified, no report will be sent out.
Mirroring.ErrorRetryPeriod	Indicates in seconds, how often (in period) to retry to handle an MR.
Mirroring.ErrorRetryTimeout	Indicates the maximum time in seconds to keep an MR in the Mirroring Queue before finally giving up mirroring it.

Mirroring.Source.ServerList	For each source specified, this defines the list of nodes in the SC to be contacted. This must be given in the format: "<node>:<port>,...".
Mirroring.Source.Period	Gives the period (sampling rate) for which to query the SC for new MR's to be handled. If not specified, the TC will not send period requests for file lists referencing new files archived in the SC.
Mirroring.Source.CompleteSync	Time (or list of time references) for which to carry out a complete synchronization with the SC. If not specified, a complete synchronization will not be carried out.
Mirroring.Source.SyncType	Gives the type of mirroring to use. Defined is: ***INGESTION_DATE***: Mirroring based on ingestion date of the Data Object (ngas_files.ingestion_date). Only files with an ingestion date after the given date are considered. ***FILTER_PLUGIN***: Apply a Mirroring Filter Plug-In on the contacted node in the SC which will generate the list of data object to mirror. It is possible to specify to use both, i.e., ingestion date in conjunction with a Filter Plug-In.
Mirroring.Source.TargetNodes	Indicates the list of target nodes to which data will be mirrored (re-archived) in the TC. The following values are defined: ***ALL***: All archiving units online in the TC cluster with available space will be considered. ***<target node 1>:<port>,<target node 2>:<port>***: The nodes listed will be used as target nodes. Note it is in principle possible to specify nodes, which are not part of the actual SC using this option.
Mirroring.Source.LocalCopies	Indicates the desired number of local copies of each file in the TC. Only if a file is available in less copies than the given, desired number of copies, it will be scheduled for mirroring.
Mirroring.Source.FilterPlugIn	Filter Plug-In invoked on the contacted SC node to determine which files to be mirrored.
Mirroring.Source.FilterPlugInPars	Parameters for the Filter Plug-In.

An example of a configuration definition:

```
<Mirroring  Active="1"
            Threads="2"
            CleanUpTime="22:30:00"
            Report="ngas-support@xxx.org,ngas-operator@xxx.org"
            ErrorRetryPeriod="30"
            ErrorRetryTimeOut="100">

  <MirroringSource ServerList="nau_1-cluster_1.xxx.org:7001,nau_2-cluster_1.xxx.org:7001"
                   Period="10"
                   CompleteSync="00:00:00,12:00:00"
                   SyncType="INGESTION_DATE"
                   TargetNodes="ncu_1-cluster_a:8001,ncu_2-cluster_a:8001"/>

  <MirroringSource ServerList="nau_1-cluster_2.xxx.org:7001,nau_2-cluster_2.xxx.org:7001"
                   Period="10"
                   CompleteSync="00:00:00,12:00:00"
                   SyncType="INGESTION_DATE"
                   TargetNodes="ncu_1-cluster_a:8001,ncu_2-cluster_a:8001"/>

</Mirroring>
```

1.36 EXPERT: DB

In the NGAS DB a table, NGAS Mirroring Queue ("ngas_mirroring_queue") is used to schedule files for mirroring and used to get a status of the mirroring. This table is described in more details in chapter 8.

1.37 EXPERT: Classes

To handle an MR, a class is provided, "ngamsMirroringRequest", which has the following attributes:

Attribute	*Description*
File ID	The NGAS File ID of the data object to be mirrored.

File Version	The NGAS File Version of the data object to be mirrored.
Ingestion Date	The ingestion date registered for the file in the SC.
Server List ID	Reference to table (ngas_mirroring_servers) containing the Server Lists defined.
XML File Info	The XML file info document for that file. This is kept in order to be able to restore the record in the TC DB after the file has been mirrored.
Status	The status of the request is one of the following: 1. **Scheduled**: The MR has been scheduled by the Mirroring Control Thread, but is not yet being handled. 2. **Active**: The MR is currently being handled. 3. **Mirrored**: The MR has been successfully mirrored. 4. **Reported**: The entry has been reported via Email Notification. 5. **Error/Abandoned**: The attempt to mirror the data object failed. It is not possible to recover from the error. 6. **Error/Retry**: The attempt to mirror the object failed. It should be retried later to handle the request.
Message	Will indicate the status of the Mirroring Request if relevant. In particular when the request is in Error State, this attribute should be set to provide diagnostics.
Last Activity Time	Gives the time (seconds since epoch) for the last time it was tried to handle the request.

1.38 EXPERT: Cache Archive Service

The NGAS Cache Archive Service is used to deploy NGAS Archives, where the data should only be kept in the archive for a certain period of time until a criterion for removing the data from the cache is fulfilled.

While the data is kept in the cache, it is kept safe as though it was archived in a normal NGAS Archive System. This also means that the data can be retrieved, processed and checked as normal data. The Data Consistency Checking will also be run on the cached data, if the unit is configured correspondingly.

.0.1 EXPERT: Basic Functioning

To distinguish clearly between operating the NG/AMS Server as a normal, safe archive service and operating it in the less safe mode, as a cache, to use an NGAS instance as a Cache Archive, a special implementation of the NG/AMS Server should be invoked. This is named “ngamsCacheServer”; the server is invoked with the same parameters as the normal NG/AMS Server:

```
$ ngamsCacheServer -cfg <Configuration> ... <Other Options> ...
```

Operating this instance of the NG/AMS Server, the Cache Service is always active per definition and will behave according to the configuration, read from the NG/AMS Configuration; see below. I.e., it is not possible to deactivate the Cache Service when running the NG/AMS Cache Archive Server.

	*The NGAS Cache Archive service should be used with great care as deploying an NGAS Node as a cache archive, may result in loss of data, possibly unintentionally. This is also the reason why a dedicated binary is provided to run a node in cache mode. Nevertheless, operating a node in cache mode means that clients may connect and archive data onto these. It might be advantageous to design the system such that data to be stored on a long-term basis are not archived unintentionally onto NGAS Cache Nodes.*

The managing of the contents of an NGAS Cache Archive is handled per node, normally per NAU. I.e., each node constituting the NGAS Cache Archive, is managing the cached files on each specific node, independently of the other nodes in the Cache Archive Cluster. I.e., each node is acting autonomously when deciding to remove data from the cache and there is no global coordination of the contents in an NGAS Cache Archive Cluster.

A Data Object archived in the cache, is identified by its File ID together with its File Version. The Disk IDs in connection with the various copies of the Data Object is not used for determining which data to keep and which to delete from the cache.

When the conditions for removing a data object from the cache are fulfilled, all copies of the given Data Object, hosted within as given NGAS Instance, are deleted from the data holding managed by that instance. I.e., if a Main and a Replication Copy are stored on the node, both instances are removed from the cache when the condition for removal is fulfilled.

.0.2 EXPERT: Caching Schemes

The handling of the data in the cache can be set up to be controlled by the following criteria:

Scheme	*Description*
Time	A data object is kept in the cache until it has been kept for certain period of time.
Volume	A certain volume of data (measured in MB) is kept. When the cache holding exceeds that size, files are removed FIFO-wise.
Number of Files	A certain number of files are kept in the cache. When the number of files in the cache exceeds this number, files are removed FIFO-wise.
Cache Storage Availability	A minimum free cache storage space is defined. If the available space in the cache drops under this level, files are removed from the cache FIFO-wise, until the desirable threshold is reached.
Cache Control Plug-In	The Cache Control Plug-In makes it possible to implement a context specific decision making of when to delete files from the cache. The Cache Control Plug-In may contact other (NGAS) Archives, when deciding whether to remove data or not.

It is possible to deploy all five schemes in conjunction, the conditions will be OR'ed in this case such that, working its way through the list according to the sequence above, at the first positive result (TRUE), the check stops and the file being checked is scheduled for deletion.

It should be noted that when a file is deleted from the cache, the entry in the NGAS Cache Table is deleted along with the entry in the NGAS File Tables, as well as the file itself. Also the entries in the DB Snapshots on the volumes concerned are removed.

.0.3 EXPERT: Configuration

The properties for operating an NGAS Cache Node, is defined in the Caching Element in the NG/AMS Configuration. The attributes of the Caching Element are defined as follows:

Attribute	*Description*
Caching.Period	Period with which the Cache Control Thread should be running its checks, to identify if files should be removed.
Caching.MaxTime	Maximum period of time during which to keep cached objects.
Caching.MaxVolume	Maximum size (in bytes) of the cache.
Caching.MaxFiles	Maximum number of files to keep in the cache.
Caching.MinCacheSize	Minimum available space requested in the cache. *Note, this feature is currently not supported*
Caching.CacheControlPlugIn	Plug-in invoked to provide context specific control of when files should be deleted from the cache.
Caching.CacheControlPlugInPars	Optional parameters for the Cache Control Plug-In.
Caching.NumberOfPlugIns	Specifies a number of helper plug-ins, to be executed to actually execute the Cache Control Plug-In on each Data Object in the cache holding. It is necessary to parallelize this processing, as in case of a high throughput in terms of files/s, the Cache Service would not be able to keep up with the incoming stream of data and the cache storage availability, would eventually saturate. Specifying the appropriate number of helper threads has to be done on an experimental basis, such that it should be tried to run the Cache Archive Node, with the worst-case (maximum) incoming data rate (measured in files/s) for a while. The cache holding should be monitored and the number of threads adjusted such that after a given period of time the number of files in the cache will fluctuate around a center value, but will stop growing further.

If a condition should be disregarded whilst operating the cache, this should be left empty.

An example of a configuration of an NGAS Cache Archive Node is as follows:

```
<Caching  Period="10"
          MaxTime="3600"
```

```
        MaxCacheSize="1000000000"
        MaxFiles="10000"
        MinCacheSpace=""
        CacheControlPlugIn="ngamsExampleCacheCtrlPI"
        CacheControlPlugInPars="max_cache_time=3000"
        NumberOfPlugIns="5"/>
```

Figure 15: Example of cache archive set-up.

This Caching Element specifies a Cache Archive Instance, which:

1. Checks the cache holding every 10s (Period=10).
2. Deletes entries from the cache, residing more than 1 hour in the cache (MaxTime=3600).
3. Ensures the size of the cache does not grow above 1 GB (MaxCacheSize=1E9).
4. Maintains the maximum number of files in the case to ten thousand (MaxFiles=1E5).
5. Executes the Cache Control Plug-In, ngamsExampleCacheCtrlPI, which implements a simple business logic, which ensures a maximum caching time, here 3000s (corresponds to MaxTime=3000). In total 5 helper thread are executed to carry out the Cache Control Plug-In check of the cache holding.

.0.4 EXPERT: Deleting Cached Data on Demand

In order to delete entries in the cache explicitly, the CACHEDEL Command should be used. The CACHEDEL Command is described in 1.80.

.0.5 EXPERT: DB

The internal housekeeping is done, using an internal SQLite based DB (Cache DBMS Table), which is located within the NGAS home directory (Root Directory). This SQLite DBMS is named:

`/NGAS/cache/CACHE_CONTENTS_DBMS_<Host>:<Port>.db`

Usually the user does not have to be concerned with this DB; possibly only for troubleshooting purposes it might be useful to peek into the internals of the Cache Service.

In order to publish the contents of the cache, a table is contained in the NGAS DB, ”ngas_cache” (RDBMS Cache Table), which reflects the contents and status of the cache. This table is defined as follows:

Attribute	*Type*	*Description*
file_id	varchar(64)	The NGAS File ID allocated to the file
file_version	int	The NGAS File Version allocated to the file
cache_time	varchar(32)	The time the file was cached.
cache_delete	tinyint	If set to “1” the file is scheduled for deletion.

The internal 'cache table' has a similar definition, but contains more information, needed for managing the cache.

It is possible to delete the internal Cache DB and the contents of the RDBMS Cache Table. The cache service will reconstruct itself from the information about the data objects archived on that NGAS Instance, contained in the NGAS Files Table (“ngas_files”). Nonetheless, if both Cache Tables are lost/contents deleted, some information will be lost like last time for checking a cache entry for deletion. This however is not a major issue and usually simply means that some entries are kept for a longer period of time in the cache. If available space in the cache is a concern, it is always possible to specify a threshold for the maximum amount of data (in entries or volume) to ensure the cache does not saturate. If data objects were scheduled explicitly for removal from the cache via the CACHEDEL Command, this information, however, is lost in case both tables are lost. In that case it will be necessary to reschedule the data objects in question for deletion. Normally this is probably not a major concern, as losing both tables should be a rare event.

Note, if the contents of the RDBMS Cache Table is deleted, but the internal DBMS Cache Table is kept, the RDBMS Cache Table will be reconstructed from the contents in the Internal DBMS Cache Table correlated with the contents in the NGAS Files Table, what concerns the data holding on that NGAS Instance. The same is true if the DBMS Cache Table is deleted and the RDBMS Cache Table preserved.

At any rate, what is described above indicates, that the Cache Service is fairly robust towards inappropriate manipulation/corruption of its persistent housekeeping information.

1.39 EXPERT: User Service Thread

The User Service Thread can be used to execute tasks in background, which needs to be carried out periodically, to e.g. keep an NGAS Node in a good condition (cleaning up) or to do periodic system checking.

The actions to be carried out should be implemented in a so-called User Service Thread Plug-In.

TODO$$$$$

5. The NG/AMS Server and Utilities

Three 'executables' are provided within the NG/AMS package. These are 1) The NG/AMS Server - "ngamsServer(.py)", 2) The NG/AMS Python Client - "ngamsPClient(.py)", 3) The NG/AMS C Client - "ngamsCClient", 4) The NG/AMS Archive Client - “ngamsArchiveClient” and 5) A CRC32 checksum utility – “ngamsCrc32”. These executables are described in the following sections.

1.40 NG/AMS Server Command Line Interface

The NG/AMS Server is the central component of the NGAS system. It is an HTTP based server/daemon, which runs on each NGAS Nodes and controls all activities with respect to the data on that node. It is also aware of other nodes and data of an NGAS System and can interact with other NGAS Nodes in the system.

By calling the NG/AMS Server without command line parameters (or illegal ones), online help is displayed on “stdout”. This can be viewed by following the link:

http://<Host>:<Port>/ngams.ngamsServer.ngamsServer_doc.html

1.41 Python and C Client Utilities

The Python and C Client Utilities are used to interact with the NGAS System from the UNIX/Linux shell. With this it is possible to send commands and receive replies to/from an NG/AMS Server specified. The contacted NG/AMS Server may in turn act as proxy and contact other NGAS Nodes on behalf of the client tool. The client tools can also handle HTTP redirection response transparently for the user.

By invoking the NG/AMS Python and C command utilities without input parameters (or with illegal ones), the following online help is written on “stdout”. This can also be viewed online by following the link:

http://<Host>:<Port>/ngams.ngamsCClient.ngamsCClient.html

1.42 The NG/AMS Archive Client

The NG/AMS Archive Client is used to archive data from a remote location in an easy and safe manner. It is very simple to set up an archiving scenario by means of the client, which archives data from any location, from which it is possible to interact with a given NG/AMS Server via the HTTP protocol used by NG/AMS.

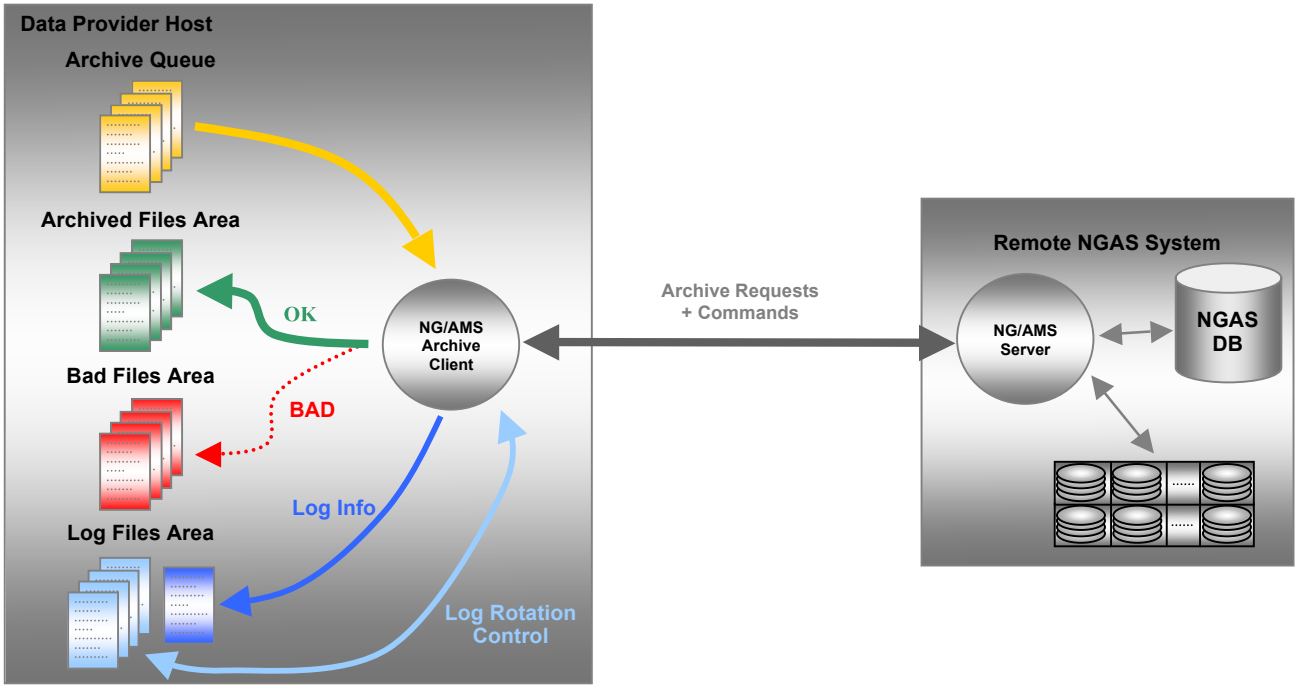

Figure 16: The NG/AMS Archive Client.

The Archive Client is normally running as a daemon in background. It communicates with the remote NGAS System, which is specified via the command line parameters of the Archive Client. The client maintains four data areas:

Area	*Description*
Archive Queue	In this folder files to be archived must be placed by the file producer application(s). It is possible also to create a link to the original copy of the file in this directory. The Archive Client polls cyclically the queue to check if there are files to be archived. If the Archive Client does not succeed in archiving a file, the file remains in the Archive Queue and it will be retried periodically to archive it.  *It should be considered carefully whether the best approach is to use a link as opposed to copy the file into the Archive Queue. Using a link saves disk space, but leaves it up to the data provider application to delete the file. This should not happen, of course, before the file has been properly archived. The NG/AMS Archive Client keeps the archived files for a while, and double-checks if each file has been properly archived in the remote NGAS System before deleting. If disk space is not an issue, it is therefore preferable to move the files to be archived into the Archive Queue rather than using links.*
Archive Files Area	This area contains the files that have been successfully archived. If the file was copied into the Archive Queue this directory will contain this 'physical' copy. If a link was created from the original copy of the file into the Archive Queue, only the link will be moved. Apart from the file/link itself, there will also be an XML document for each file archived containing the status returned from the remote NGAS System. This area is cleaned when the files have resided there for a specified period of time. Before deleting any archived file, the Archive Client checks if the file is available in the remote NGAS System. This is done by sending a CHECKFILE Command (section 1.81) to the remote NGAS System, which subsequently carries out a consistency check of the file.
Bad Files Area	If a file is classified as bad by the remote NGAS System, the copy of link of the file is moved to the Bad Files Area. An XML document with the status from the remote NGAS System is created together with the file/link. This indicates which kind of problem was encountered with the file.  *Note, the NG/AMS Archive Client does not clean up any files in the Bad Files Area. It is the responsibility of the operators of the site where the Archive Client is running, to monitor that the amount of data in the Bad Files Area does not saturate the system.*
Log Files Area	The Archive Client logs information into a log file according to the Log Level specified. The Archive Client can rotate the current log file on a daily basis, at a given time of the day. It is possible to specify the history (= number of log files to keep). When a log file is rotated, it is possible to specify to the Archive Client that it should archive the file into the remote NGAS System. This requires that the NGAS System is configured for handling this type of data.

The NG/AMS Archive Client has no direct interaction with the NGAS DB. All information needed from the DB is requested from the remote NGAS System, which has DB access. This makes the Archive Client simpler. This also means that the installation and configuration of the NG/AMS Archive Client is quite simple.

The amount of disk space on the host where the Archive Client is running, should be carefully dimensioned such that the data rate of the data providers fits together with the amount of buffering capacity of files in the Archive Queue, the time for keeping data in the Archived Files Queue, the amount of Bad Files that should be buffered and the amount of log files. Usually it would probably make sense to be able to buffer some days of data.

It would probably make sense to implement a small supervision tool, which is executed on a daily basis (as a cron job). It should monitors the amount of free disk space and issue a warning if this goes below a certain limit. This script could also inform the operators about files in the Bad Files Area, which needs to be handled.

By invoking the client on the shell without input parameters, the man-page is displayed. The man-page can also be accessed online by following the link:

http://<Host>:<Port>/ngams.ngamsCClient.ngamsArchiveClient.html

1.43 NG/AMS CRC-32 Utility

The NG/AMS project provides a small utility to calculate the CRC-32 of a given data file. This is used internally in NG/AMS also to calculate the checksum for files when these are archived. This utility is called “ngamsCrc32” and is provided by the NG/AMS C-Client Module.

6. *EXPERT:* Configuring NG/AMS

The NG/AMS SW is implemented in a very flexible way to be able to adjust the system for various scenarios. In order to obtain this, a wide range of parameters can be adjusted in the NG/AMS Configuration.

This chapter contains a description of these parameters. The format for the NG/AMS Configuration also includes the Header Element which is a generic standard header for XML documents. This header is not described here. An example of an NG/AMS Configuration can be found in section 1.47. The DTDs defining the format of the NG/AMS Configuration can be found in the sections 1.45 and 1.46.

Note, from NG/AMS V2.3 it is possible to store the configuration in the NGAS DB. The NG/AMS Servers then loads the configuration from a central place. How to handle this is described in section 1.44.

1.44 EXPERT: Handling the NG/AMS Configuration in the NGAS DB

From V2.3 of NG/AMS it is possible to load XML configuration files into the DB and to request that the NG/AMS load the configuration from there rather than from a file based document. The configurations are contained in the tables “ngas_cfg” and “ngas_cfg_pars” (section 1.51).

All the parameters of the various configurations are contained in “ngas_cfg_pars”. In connection with each element in the configuration, it is possible to define an attribute, “Id”. This defines the group to which the parameters in that element belong. The last “Id” attribute encountered during parsing in the XML document determines the group of each parameter. This ID is referred to as DB Configuration Group ID.

In the “ngas_cfg” table it is possible to build up configuration by combining the defined group IDs in the “ngas_cfg_pars” table. This is then allocated a DB Configuration ID. The NG/AMS Server must be started with the DB Configuration ID. The parameters defined for that ID will then be loaded into the server.

It is possible to split configuration files into ‘partial configuration documents’

The parameters of XML configurations can be loaded using the “ngams/ngamsLib/ngamsConfig.py” module. This must then be instructed to load the configuration into the DB. An example of this is shown in the following:

```
[ngasmgr@ngasdev2]$ python /opsw/packages/ngams/ngamsLib/ngamsConfig.py -cfg
/opsw/packages/ngasCfg/cfg/NgamsCfg.PERM-AHU.xml -storeDb "TESTSRV/ngaststl/ngas/*****"
Loading configuration: /opsw/packages/ngasCfg/cfg/NgamsCfg.PERM-AHU.xml
Configuration: /opsw/packages/ngasCfg/cfg/NgamsCfg.PERM-AHU.xml loaded!
Loading configuration into the DB ...
Loaded configuration into the DB
[ngasmgr@ngasdev2]$
```

Figure 17: Example of how to load an XML configuration into the DB.

Note, the NG/AMS Server must always be started with a valid XML configuration. This however, may contain only the definition of the DB connection (CFG: “Ngams.Db”).

There is not yet any tool to view and edit the configuration parameters graphically.

1.45 EXPERT: NG/AMS Configuration DTD - "ngamsCfg.dtd"

The DTD for the NG/AMS Configuration is based on the "ngamsInternal.dtd" (see section 1.46), which defines the NG/AMS specific elements used in the NG/AMS Configuration. The contents can be viewed following the link:

http://<Host>:<Port>/ngams.ngamsData.ngamsCfg_dtd.html

1.46 EXPERT: NG/AMS Base DTD - "ngamsInternal.dtd"

The base DTD is used to define various XML elements, which can be re-used in various deducted DTD/XML documents. The contents can be viewed following the link:

http://<Host>:<Port>/ngams.ngamsData.ngamsInternal_dtd.html

It is planned at a later stage to switch to use XML Schema rather than DTD.

1.47 EXPERT: NG/AMS Configuration – Examples

An example of a complete/stand-alone configuration file for Simulation Mode can be viewed by following the link:

http://<Host>:<Port>/ngams.ngamsData.ngamsCfgSim_xml.html

Examples of configuration files used for operation can be found following the link:

http:// <Host>:<Port>/ngasCfg.cfg.html

These XML document only contains parts of XML configurations. These are used to build up configurations in the NGAS DB, which can be loaded by the NG/AMS Server when going Online.

7. *EXPERT:* NG/AMS Server Communication Protocol

The NG/AMS command interface is based on the HTTP protocol, which is a widely used standard protocol. This makes it easy to interface various client applications with NG/AMS. In this chapter the details of the NG/AMS command interface are described.

Using the NG/AMS Python- and C-APIs, the client applications do not need to worry about the format of the requests sent and replies generated by NG/AMS. It is therefore recommended whenever possible to use the APIs provided with the NG/AMS package.

1.48 EXPERT: Format of NG/AMS HTTP Command Messages

The format of the NG/AMS messages is defined as follows:

Archive Push Request:

```
POST ARCHIVE HTTP/1.0
User-Agent: <user agent>
Content-Type: ngas/archive-request | <mime-type>
Content-Length: <length>
Content-Disposition: attachment; filename=<file uri>[; wait=0|1][; no_versioning=0|1]

<data>
```

Figure 18: Format of an Archive Push HTTP request.

Example:

```
POST ARCHIVE HTTP/1.0
User-Agent: NG/AMS C-API
Content-Type: ngas/archive-request
Content-Length: 69120
Content-Disposition: attachment; filename="/tmp/TestFile.fits";wait="1"

~   ‡v}yzy~ƒ}{uˆ~,‰…tcv,, …
```

Figure 19: Example of Archive Push HTTP request.

Archive Pull Request + Other Commands:

```
GET <command>?[<parameter>=<value>] HTTP/1.0
```

Figure 20: Structure of NG/AMS GET method HTTP request.

Example, Archive Pull Request:

```
GET ARCHIVE?filename="file:///tmp/SmallFile.fits"&wait="1" HTTP/1.0
```

Figure 21: Example of NG/AMS GET method HTTP request (Archive Pull Request).

The exact list of parameters for each command is described in chapter 16.

1.49 EXPERT: Format of the NG/AMS HTTP Reply

The format of replies from NG/AMS is defined as follows:

```
HTTP/<HTTP version> <HTTP response code> <message>
Server: <server ID>
Date: <date for generating reply>
Expires: <expiration date (= Date:)>
Content-Type: <mime-type>
Content-Length: <data length>

<data>
```

Figure 22: Format of NG/AMS HTTP response.

An example of a reply to an Archive Request is:

```
HTTP/1.0 200 OK
Server: NGAMS/v2.0-Beta2/2002-12-04T09:22:53
Content-type: text/xml
Expires: Mon, 23 Dec 2002 16:10:43 GMT
Content-length: 1188
Date: Mon, 23 Dec 2002 16:10:43 GMT

<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-12-23T16:10:43.079" HostId="acngast1" Message="Successfully handled Archive Push
          Request for data file with URI: SmallFile.fits" State="ONLINE" Status="SUCCESS"
          SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
  <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32300" BytesStored="8567866905" Checksum=""
              Completed="0" CompletionDate="" DiskId="IC35L040AVER07-0-SXPTX093675" HostId="acngast1"
              InstallationDate="2002-11-25T09:48:25.000" LastCheck="" LogicalName="FITS-M-000001"
              Manufacturer="IBM" MountPoint="/NGAS/data1" Mounted="1" NumberOfFiles="164" SlotId="1"
              TotalDiskWriteTime="896.20280099" Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1246906309" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="TEST.2001-05-08T15:25:00.123"
                FileName="saf/2001-05-08/3/TEST.2001-05-08T15:25:00.123.fits.Z" FileSize="53546"
                FileStatus="00000000" FileVersion="3" Format="application/x-cfits" Ignore="0"
                IngestionDate="2002-12-23T16:10:42.000" Tag="" UncompressedFileSize="69120"/>
  </DiskStatus>
</NgamsStatus>
```

Figure 23: Example of NG/AMS HTTP response (Archive Request).

In a reply to a Retrieve Request the data returned will be contained in the message rather than the NG/AMS Status XML document shown above. Such a reply thus looks like this, e.g.:

```
HTTP/1.0 200 OK
Server: NGAMS/v2.0-Beta2/2002-12-04T09:22:53
Content-type: application/x-cfits
Expires: Mon, 23 Dec 2002 16:15:22 GMT
Date: Mon, 23 Dec 2002 16:15:22 GMT
Content-disposition: attachment; filename="TEST.2001-05-08T15:25:00.123.fits.Z"
Content-length: 53546

<data>
```

Figure 24: Example of NG/AMS HTTP response, Retrieve Request.

It is foreseen at a later stage to make it possible to query several files simultaneously with one query. This means that the mime-type "multipart/mixed" will be used as the overall mime-type of the reply and that each part has its proper mime-type defined.

1.50 EXPERT: Format of the NG/AMS HTTP Redirection Response

If an NG/AMS Server is not configured to always act as a proxy when data is being requested by a client, HTTP redirection response messages may be generated and send back to the requestor. The format of such redirection responses is:

```
HTTP/1.0 303 OK
Server: <server ID>
Date: <date>
Expires: <date>
Location: <URL pointing to actual location of file>
Content-Type: text/xml
Content-Length: <length>

<NG/AMS status document>
```

Figure 25: Structure of NG/AMS HTTP Redirection Response.

An example of such a redirection HTTP response is:

```
HTTP/1.0 303 Method
Server: v1.5/2002-02-12T10:52:10
Date: Tue,  Jan 01:34:40 2 GMT
Expires: Tue,  Jan 01:34:40 2 GMT
Location: http://jewel64:7777/RETRIEVE?file_id="WFI.2001-09-25T21:19:17.508"
Content-Type: text/xml
Content-Length: 339
```

```
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2001-01-02T01:34:40.656" HostId="jewel68"
          Message="NGAMS_INFO_RETRIEVE_REDIRECT:4024:INFO:
          Redirection URL: http://jewel64:7777/RETRIEVE?file_id=WFI.2001-09-25T21:19:17.508"
          State="ONLINE" Status="SUCCESS" SubState="BUSY" Version="v1.5/2002-02-12T10:52:10"/>
</NgamsStatus>
```

Figure 26: Example of NG/AMS HTTP Redirection Response.

The client must then re-issue the Retrieve Request to the alternative location given in the redirection response and will be able to get access to the data directly from that location (if the system permits). It should be mentioned that it is normally more efficient to request the data directly from the location where it is actually located rather than using NG/AMS as a proxy server. Using the NG/AMS APIs this is all handled transparently for the client application.

8. *EXPERT:* The NGAS DB

The NGAS DB has been designed to be as simple as possible. This means that all the business logic of the system is contained in the code and not in stored procedures etc. in the DB. This has the advantage of making it easier to adapt the systems for usage with other DBMS'. In this chapter the details of the NGAS DB are described what concerns the lay-out of the DB and synchronization of various, remote NGAS DB's

1.51 EXPERT: Layout of the NGAS DB

The NG/AMS SW is based on 7 tables in the NGAS DB. These are:

Table	*Description*
ngas_cfg	Contains definition of the configurations to be used by the various NGAS systems in operations. These configurations are defined as an ID with an associated set of Configuration Group IDs, referring to groups of parameters in the "ngas_cfg_pars" table.
ngas_cfg_pars	Used to store the configuration parameters. Each parameter has an associated Configuration Group ID. The parameter name and value and an optional comment is stored for each parameter. The parameter name is given in the XML Dictionary Key Format. All parameter values are given as strings.
ngas_disks	Contains information about the disks, which have been registered in an NGAS installation.
ngas_disks_hist	Contains a log about major events that has occurred in the life-time of a disk. The information in the "ngas_disks_hist" is newer removed automatically by NG/AMS and will continue to grow with time, whereby the amount of events recorded are kept to a minimum, i.e., only the most essential events in the life-time of a disk are logged in this table. The entries of this table can be used to keep analyze how a disk have been used in the NGAS system, e.g. how many times it has been registered (after 're-cycling'), and when it was registered the first time.
ngas_files	Contains information about each files, which have been archived into NGAS
ngas_ops_log_book	The Operators Log Book is used by the NGAS Operators to keep track of the various actions performed. In this way it is possible to trace the actions performed in connection with a specific component, e.g. a disk.
ngas_hosts	Contains information about the hosts in an NGAS installation.

To handle the Subscription Service the following two tables are used:

Table	*Description*
ngas_subscr_back_log	Used by NG/AMS to keep track of files that should have been delivered to a Subscriber whereby the delivery failed.
ngas_subscribers	Contains a persistent snap-shot of the Subscribers that are subscribed to a given NG/AMS Server.

The NGAS DB lay-out is fairly simple. This combined with the NGAS DB Driver Plug-In, makes it possible to use NGAS together with other DB systems. NGAS has been used together with Sybase, Oracle, MySQL and SQLite.

To handle the Mirroring Service the following two tables are used:

Table	*Description*
ngas_mirroring_queue	Used by the NGAS Mirroring Service to keep track of the files scheduled for mirroring. The files are referred to by their File ID and Version. The status of the mirroring request is contained in the table, together with information about scheduling time and last time there was activity in connection with that request.
ngas_mirroring_hist	In this table, a history of handled mirroring requests is kept for a certain period of time as a status.

Usually it is not foreseen that external applications perform queries directly into the NGAS DB. I.e., all information needed, should be retrieved via the NG/AMS Server. Apart from saving external applications from knowing technical details about the NGAS DB, this has the advantage of making such external applications independent of the DBMS used by an NG/AMS installation. For this reason, it is not guaranteed that 100% backwards compatibility is maintained when it comes to the format of the NGAS DB.

The description of the tables in the NGAS DB and each column in these, can be obtained by following the link:

http://<Host>:<Port>/ngams.ngamsSql.html

1.52 EXPERT: Internal DB Synchronization/DB Snapshot Feature

Since disks may travel between NGAS Sites and thus using independent DBMS', NG/AMS implements an internal scheme for synchronization such remote NGAS DB's. This is done by means of the so-called DB Snapshot, which is a file based DB stored on each disk (DBM/BSDDB). This is updated when the NG/AMS Server starts up and when changes are introduced in the data holding for each disk. See section 1.12.

It is only the information about files that is synchronized, i.e., the information in the "ngas_files" table. The information in the DB Snapshot is packed in a binary format for performance reasons.

The DB synchronization/DB Snapshot update is handled in the following way:

- When the NG/AMS Server starts up it checks if the DB Snapshot is up-to-date compared to the NGAS DB. If the DB Snapshot does not exist at this point, it is created.
- If files are found in the NGAS DB, which are not registered in the DB Snapshot, they are added in the DB Snapshot if the file is also found physically on the disk. If the file is not found on the disk, it is deleted from the NGAS DB.
- If files are found in the DB Snapshot, which are not registered in the NGAS DB, they are updated in the NGAS DB if physically found on the disk. If they are not found on the disk, the entries are deleted from the DB Snapshot.
- If entries for files are found in the NGAS DB and the DB Snapshot, and the file is found on the disk nothing is done. If the file is not found on the disk, a Data Check Notification Email is sent out indicating that files are lost:

```
JANITOR THREAD - LOST FILES DETECTED:

==Summary:

Date:                       2004-08-10T12:07:43.350
NGAS Host ID:               ngahu
Lost Files:                 1

==File List:

Disk ID             File ID                    File Version Expected Path
------------------- -------------------------- ------------ ------------------------------------------------------------------------
WDC-WD-WMAEH1993650 WFI.1999-10-19T06:37:09.854 1            /NGAS/data2/saf/1999-10-18/1/WFI.1999-10-19T06:37:09.854.fits.Z

==END
```

Figure 27: Example of a Lost File Email Notification Message from the Janitor Thread.

- During operation of the NG/AMS Server, the DB Snapshot is updated 'real-time' each time a modification related to a file is introduced in the NGAS DB.

Note, no files are ever deleted by the DB Snapshot feature.

	The NG/AMS DB Snapshot Feature makes it possible to operate completely separated NGAS Sites and to freely send disks around. These and the files on them will be registered as soon as they arrive at another NGAS Site and are put Online in an NGAS Node there. After the DB Snapshot synchronization has taken place, the files on the disk are subsequently available within the context of the given NGAS Site.
	*If there are many files stored on a disk, the actual synchronization of the DB Snapshot/NGAS DB, may take a while, since this task is given a low priority in order not to disturb operations.*

1.53 EXPERT: DBMS Based Synchronization of Distributed NGAS DBs

Operating an NGAS system, which makes use of several, independent DBMS' constitutes a delicate problem in order to ensure that all sites are kept up to date, and to avoid corruption of the data holdings due to 'unforeseen/unwanted' replication between the sites.

As an example of this, it can be mentioned that if disks are prepared in the Archive Facility Site and distributed to various, remote Data Production Sites, it is desirable that 1) The DB at the Archive Facility Site is kept up to date, 2) The NGAS installation at the Production Site recognizes that the disk is already registered as an NGAS Storage Media (see also section 1.14) and 3) At the same time, after a Storage Media has been completed, has left the Production Site, and is located in an NGAS Host at the Archive Facility Site, that it is no changes introduced to the record for that disk at the remote site, are replicated to the Archive Facility Site NGAS DB.

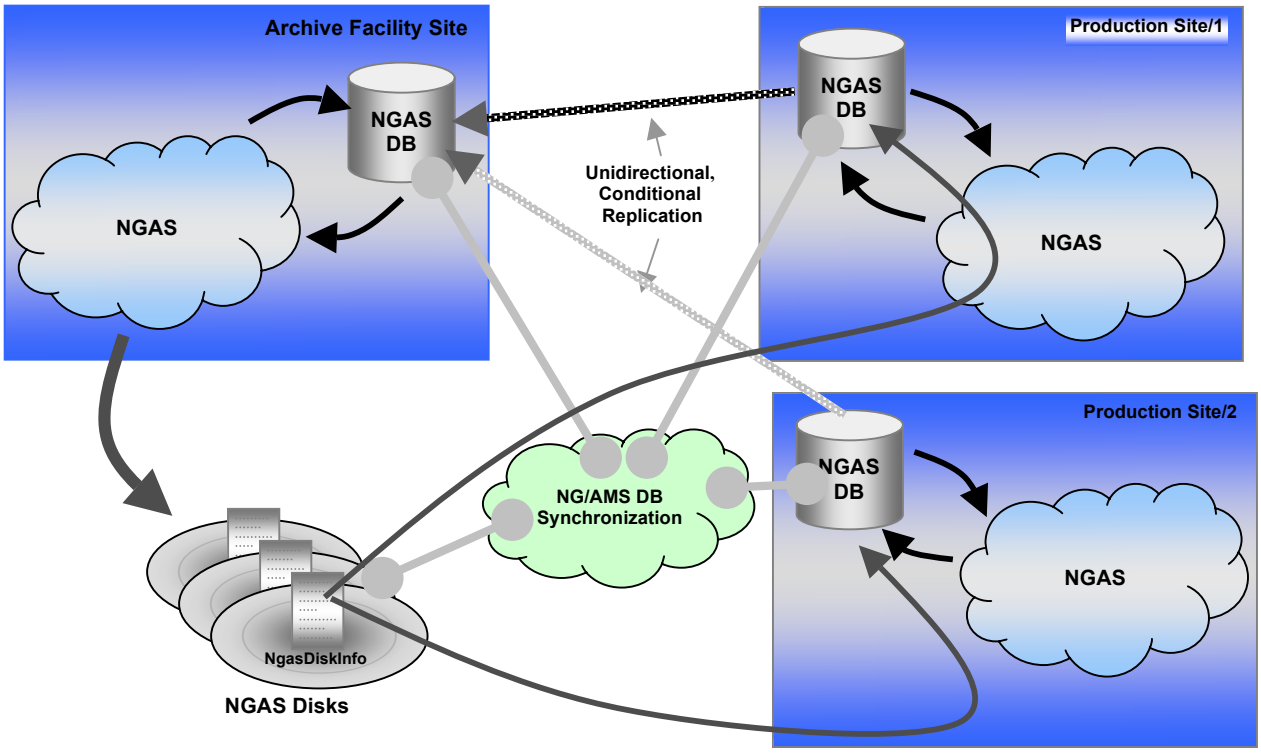

Figure 28: Example of a Distributed NGAS installation using unidirectional, conditional DB replication.

A way to implement this, is as follows:

> When NG/AMS registers a new disk, it creates an entry for this in the NGAS DB at the Archive Facility Site. NG/AMS also generates an "NgasDiskInfo" document (see section 1.73) on the disk. In this way the disk is 'marked' as a 'known' NGAS Disk, and wherever it appears, NG/AMS will recognize the disk and take the information in the "NgasDiskInfo" document and write this to the NGAS DB connected (if the disk is not already registered in that DB). This means that after having prepared the disk at the Archive Facility Site and after this has been received and installed at the Production Site, the disk has now been registered in the NGAS DB at the Production Site, based on the "NgasDiskInfo" document. The Production Site NGAS System, now archives data on the disk. During this phase, all changes in connection with the disk are replicated from the Production Site NGAS DB to the Archive Facility Site NGAS DB (disk information + information about new files). When the disk is completed, it will be marked as such by NG/AMS at the Production Site, and this information replicated to the Archive Facility Site. The disk is subsequently sent to the Archive Facility Site where NG/AMS recognizes the disk and updates the information about the disk in the DB at that site. In order to prevent that changes introduced from this point on, at the Production Site where the disk was residing, are replicated to the Archive Facility Site DB, the DB replication could implement some conditions indicating when to actually update a record in the Archive Facility DB. This could e.g. be done by using the information in the "ngas_hosts" table. If the disk for which information is received for update is located in a host at the Archive Facility Site, such an update is discarded. The replication engine can determine where the disk is located from the columns "ngas_hosts.host_id" and "ngas_hosts.domain". The complete association to do implement is as follows: "ngas_disks.host_id" → "ngas_hosts.host_id" → "ngas_hosts.domain".

Note that using *bi-directional* DB replication may not be an optimal solution either, as unforeseen/unwanted changes in the Production Site DBs is propagated to the Archive Facility Site(s). Using bi-directional replication, it would still be necessary to implement a conditional replication as explained previously.

The scenario described in this section should be seen mainly as an example, and could be used as 'inspiration' when designing the DB system for a distributed NGAS installation.

	*Note, that the NG/AMS DB Snapshot Synchronization Feature is always running in background and synchronizing the NGAS DB and the DB Snapshot on each disk. This should normally not disturb other type of replication implemented, or, the additional replication should take this into account.*

9. *EXPERT:* The C-API

Together with the NG/AMS package, an API to be used for interfacing C applications with the NG/AMS Server is provided. This is provided in the form of a small library with a set of functions making it easy to communicate from client applications to the NG/AMS Server. Also a number of various macros are provided by the C-API.

The source and the header files for the C-API are contained in the module: "ngams/ngamsCClient". This CVS module contains the files:

File	*Description*
Makefile	Make that generates/compiles the C-API. Can be invoked with the parameters clean and all, e.g.: "make clean all".
README	Documentation for this sub-module.
__init__.py	Python init script to make it possible to view the documentation for the C-Client with Pydoc.
ngams.h	Header file for the NG/AMS C-API module. This contains the definition of the function prototypes, and the definition of various macros that can be used in the clients built using the NG/AMS C-API.
ngamsArchiveClient.c	Source code for the NG/AMS Archive Client (section 1.42).
ngamsArchiveClient.h	Header file for the NG/AMS Archive Client server.
ngamsArchiveClient.doc	Documentation for the NG/AMS Archive Client.
ngamsCClient.c	The source file for the NG/AMS C based command line utility. See also section 1.41 for more information about this tool.
ngamsCClientLib.c	The source file for the library functions provided by the NG/AMS C-API.
ngamsCrc32.c	Source code for the NG/AMS CRC-32 checksum utility.

Table 13: Source files in the C-API module.

Compiling the "ngamsCClient" module, the following source files and binaries are generated:

File	*Description*
libngams.a	The library to be linked with applications using the NG/AMS C-API.
ngamsARCH_CLI_MAN_PAGE.c	The man-page for the NG/AMS Archive Client.
ngamsArchiveClient	The binary/executable of the NG/AMS Archive Client.
ngamsCClient	The binary/executable utility, which can be used to communicate with the NG/AMS Server from the command line. Refer to section 1.41 for further information.
ngamsCrc32	The NG/AMS CRC-32 utility.
ngamsLICENSE.c	Header file containing the text of the license agreement for NG/AMS (see chapter **Error! Reference source not found.**).
ngamsMAN_PAGE.c	Header file containing the text of the man-page for the NG/AMS C-API command line utility (see section 1.55).
ngamsVERSION.c	Header file containing the version information for the given distribution of NG/AMS.

Table 14: Files generated compiling the C-API.

In the following sections the header file for the NG/AMS C-API is listed. In addition the man-page for the C-API library functions is shown.

1.54 EXPERT: NG/AMS C-API - Header File: "ngams.h"

The source of the NG/AMS C-API header file can be found in the NG/AMS module as follows: "ngams/ngamsCClient/ngams.h". It contains the prototype definitions for the various functions provided by the API, and also the definition of various macros. The contents can also be viewed online by following the link:

http://<Host>:<Port>/ngams.ngamsCClient.ngams_h.html

1.55 EXPERT: NG/AMS C-API - Man Page

The information provided by the man-page for the NG/AMS C-API can be viewed by following the link:

http://<Host>:<Port>/ngams.ngamsCClient.ngamsCClientLib.html

10. *EXPERT:* The Python API

The NG/AMS Python API can be used by Python applications to interface with the NG/AMS Server in an easy and straightforward manner. The API hides most of the technical details of the NG/AMS communication interface.

To use the Python API, the following "import" statements must be contained in the client application:

```
from   ngams import *
import ngamsPClient

...
```

Figure 29: Using the NG/AMS Python-API.

The API provides a class "ngamsPClient", which is contained in the Python module "ngamsPClient.py". The complete documentation for the API is contained as in-line, Python documentation strings in the source file.

The complete documentation for the NG/AMS Python + API can be viewed by following the link:

http://<Host>:<Port>/ngams.ngamsPClient.ngamsPClient.html

A small example application based on the NG/AMS Python API is listed in the following. It is used to archive a file:

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsPClientEx.py,v 1.2 2002/02/26 17:25:41 safcvs Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------
# jknudstr  26/02/2002  Created
#
"""
Small example application archiving a file.
"""

import sys
from   ngams import *
import ngamsPClient

# Check the input parameters.
if (len(sys.argv) != 4):
    print "Correct usage is:\n"
    print "ngamsPClientEx <host> <port> <file URI>\n"
    sys.exit(1)

# Get the parameters for handling the archiving.
host    = sys.argv[1]
port    = sys.argv[2]
fileUri = sys.argv[3]

# Create instance of NG/AMS Python API.
client = ngamsPClient.ngamsPClient(host, port)

# Execute the command.
status = client.archive(fileUri)

# Handle result - here we simply print the XML status message to stdout.
print status.genXml(0, 1, 1, 1).toprettyxml('  ', '\n')[0:-1]
```

Figure 30: Small example program using the Python-API (FILE: "ngams/ngamsPClient/ngamsPClientEx").

This small test program will generate an output as the following on stdout while archiving the file (example):

```
ngasmgr@acngast1:/opsw/NGAS/ngams> python ngamsPClient/ngamsPClientEx.py acngast1 7777 /home/ngasmgr/tmp/WFI.2001-
09-15T22\:49\:07.652.fits
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-12-31T09:28:09.251" HostId="acngast1" Message="Successfully handled Archive Push Request
          for data file with URI: WFI.2001-09-15T22:49:07.652.fits" State="ONLINE" Status="SUCCESS"
          SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
  <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32300" BytesStored="8709834319" Checksum="" Completed="0"
```

```
            CompletionDate="" DiskId="IC35L040AVER07-0-SXPTX093675" HostId="acngast1"
            InstallationDate="2002-11-25T09:48:25.000" LastCheck="" LogicalName="FITS-M-000001"
            Manufacturer="IBM" MountPoint="/NGAS/data1" Mounted="1" NumberOfFiles="163" SlotId="1"
            TotalDiskWriteTime="905.324898006" Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1810827525" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
              FileId="WFI.2001-09-15T22:49:07.652"
              FileName="saf/2001-09-15/1/WFI.2001-09-15T22:49:07.652.fits.Z" FileSize="142074506"
              FileStatus="00000000" FileVersion="1" Format="application/x-cfits" Ignore="0"
              IngestionDate="2002-12-31T09:28:08.000" Tag="" UncompressedFileSize="141546240"/>
  </DiskStatus>
</NgamsStatus>
ngasmgr@acngast1:/opsw/NGAS/ngams>
```

Figure 31: Output on “stdout” from example program using the Python-API.

11. *EXPERT:* The NG/AMS Plug-Ins

NG/AMS itself, does not provide any specific support for handling of specific HW or files. All this kind of tasks is maintained by various plug-ins, which are referred to in the NG/AMS Configuration.

Programming plug-ins, may not be a straightforward task and requires knowledge of the Python programming language and of the NG/AMS project as such. Nevertheless, by means of the plug-in concept it is possible to adapt the system to operate in many different kind of environments and with some experience, this may not be so complicated after all.

This chapter describes how to implement the various types of plug-ins used by NG/AMS.

1.56 EXPERT: The NG/AMS Plug-In API

The NG/AMS Plug-In API provides convenience functions to facilitate the implementation of the various types of plug-ins used within the context of NG/AMS. The actual thorough documentation is contained as inline Python documentation strings in the code itself. For further information about this issue consult section 1.76.

It is recommended to restrict the usage of functions from NG/AMS modules to only the one ones contained in the NG/AMS Plug-In API (FILE: "ngams/ngamsLib/ngamsPlugInApi.py"). It should be mentioned, that for the moment the amount of convenience functions provided is limited. Basically, only the functions needed for implementing the plug-ins provided so far have been considered in this context. If new functions are needed requests for such can be issued to: ngast@eso.org.

The documentation for the NG/AMS Plug-In API can be viewed by following the link:

http://<Host>:<Port>/ngams.ngamsLib.ngamsPlugInApi.html

Examples of plug-in can be found in the chapters: 1.58 (System Online Plug-In), 1.59 (System Offline Plug-In), 1.60 (The Label Printer Plug-In), **1.61 (The Data Archiving Plug-In)** , 1.63 (The Data Processing Plug-In), 1.64 (The Data Checksum Plug-In) and 1.68 (The Disk Sync Plug-In).

Apart from the functions contained in the module "ngams/ngamsLib/ngamsPlugInApi.py", the following classes are used for implementing the plug-ins: "ngamsServer", "ngamsConfig", "ngamsReqProps", "ngamsDppiStatus", "ngamsDb", "ngamsPhysDiskInfo" (NG/AMS Disk Dictionary). These classes are all described in more details in chapter 13.

Frequently needed in plug-in is access to the NG/AMS Configuration and to the NGAS DB. Access to these can be obtained by means of the methods “ngamsServer.getCfg()” and “ngamsServer.getDb()”. A reference to the “ngamsServer” object is handed over to all types of NG/AMS plug-in functions.

To be able to efficiently write plug-ins for NG/AMS, it is required to have a more a less profound overview of the NG/AMS SW, or at least this will be of major advantage, depending on the complexity of the tasks performed by the plug-ins. An overview of the NG/AMS SW is given in chapter 13.

1.57 EXPERT: Transferring Client Parameters to a Plug-In

It is possible for a client application to transfer parameters to a plug-in. These may not be known to NG/AMS as such. Basically for all plug-ins described in the following, which have the “ngamsReqProps” as input parameter, it is possible to transfer client/user provided parameters.

The way this is done is simply by specifying the parameter and the corresponding value in the HTTP request as any HTTP parameter, e.g.:

http://ngas1:7878/ARCHIVE?filename=http://ngasrem1:6767/RETRIEVE?file_id=TestFile&client_par=process

The link shown is not an existing link. An NGAS Host, “ngas1:7878” is requested to archive a file, which is stored on another NGAS Host, “ngasrem1:6767”. The client archiving the file, is providing the parameter “client_par” with the value “process”. Note, the name of the parameter can be determined by the client application and must be understood by the plug-in executing the request.

With the plug-in, the parameter and the value can be accessed as follows (supposing the plug-in in question is a DAPI):

```
def MyPlugIn(srvObj,
             reqPropsObj,
             filename):
"""
Example plug-in.
"""
…

# Extract plug-in parameter if specified.
if (reqPropsObj.hasHttpPar("client_par")):
    clientPar = reqPropsObj.getHttpPar("client_par")
else:
    clientPar = None

…
```

Figure 32: How to access client provided plug-in parameters in a DAPI.

As seen in ***Figure 32***, the DAPI first checks if the client, HTTP parameter is defined. In case yes, the value is extracted and assigned to the variable “clientPar”.

Using this scheme makes the concept of plug-ins very powerful and flexible, since it is possible to provide specific information to the plug-in during operation and to make the plug-in change behavior at run-time.

1.58 EXPERT: The System Online Plug-In

The purpose of the System Online Plug-In, is to prepare the system for the Online State, where it must be fully operational according to the configuration. During this phase the storage disks are usually mounted and possibly checked for proper functioning and accessibility. A very essential task of a System Online Plug-In is to generate the so-called Physical Disk Dictionary. This contains the 'physical' information about the disks installed in an NGAS Host.

The plug-in is invoked by NG/AMS when it is going Online, i.e., either when it has received an ONLINE Command or when it has been started with the "-autoOnline" command line parameter. The actual implementation depends highly on the context (HW) in use and other specific requirements in connection with an NGAS Node. The System Online Plug-In is not executed when running in Simulation Mode (see section 1.20).

.0.6 EXPERT*:* Interface of a System Online Plug-In

The System Online Plug-In must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A System Online Plug-In has an interface as shown in ***Figure 33***.

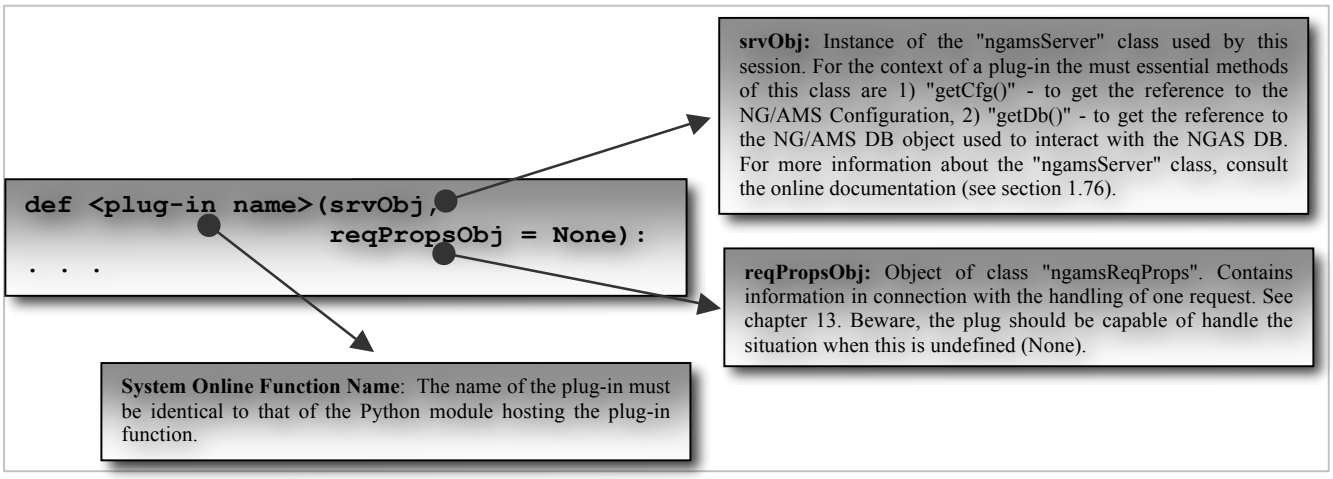

Figure 33: Function interface of a System Online Plug-In.

The return value of a System Online Plug-In is the Physical Disk Dictionary. This must be generated by the plug-in. It is a standard Python dictionary with "ngamsPhysDiskInfo" objects stored in it. The Slot IDs of the

disks are used as keys in the dictionary. The Disk Dictionary is very essential for the proper operation of NG/AMS. It is therefore crucial that the plug-in extracts and generates this information correctly for NG/AMS. The contents of the Physical Disk Dictionary is depicted in ***Figure 34***.

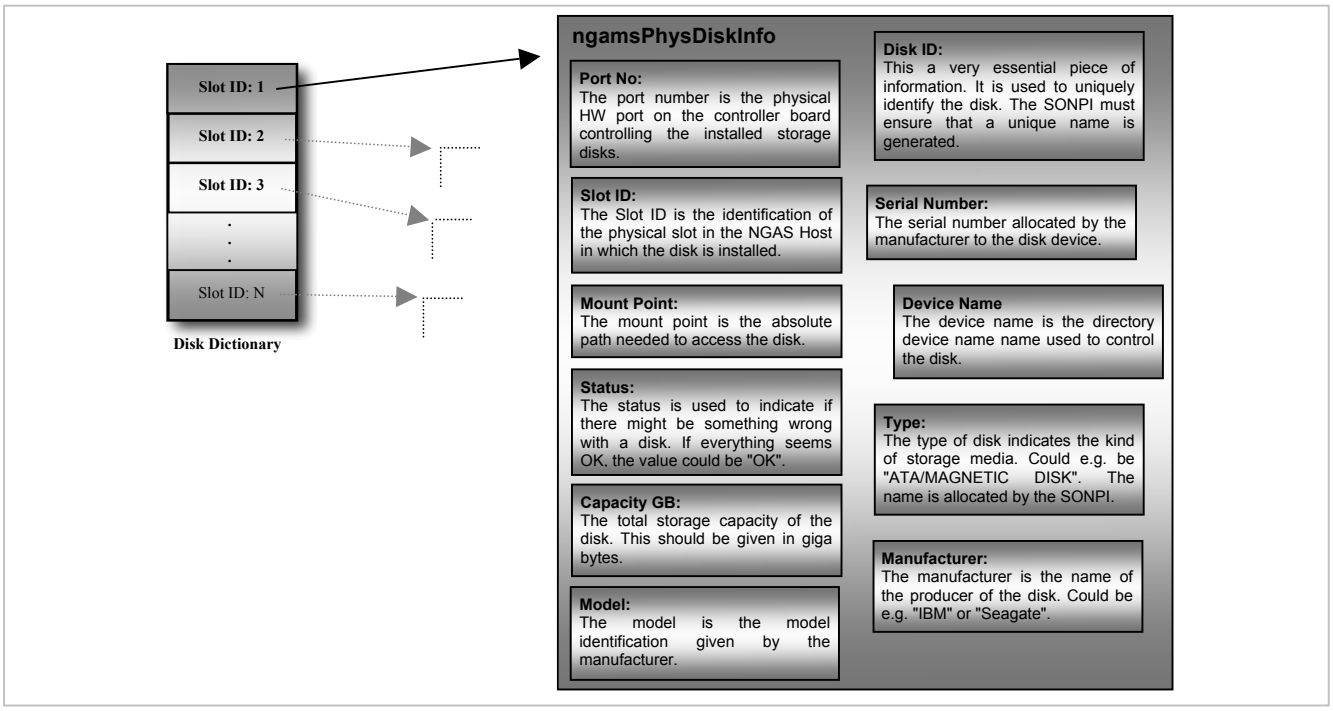

Figure 34: The NG/AMS Physical Disk Dictionary.

An exception must be thrown in case errors occur during the process of bringing the system to Online State.

.0.7 EXPERT*:* Example System Online Plug-In

In the following an example System Online Plug-In, which is used for the moment for the NGAS installation at the 3 ESO NGAS Sites. It is perhaps not a very good example of such a plug-in since most of the code is distributed in other modules. Please check the Python source files "ngams/ngamsPlugIns/ngamsEscaladeUtils.py" and "ngams/ngamsPlugIns/-ngamsLinuxSystemPlugInApi.py" for further information.

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsLinuxOnlinePlugIn.py,v 1.28 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ---------------------------------------------------------
# jknudstr  10/05/2001  Created.
#

"""
Module that contains a System Online Plug-In used by the ESO NGAS
installations.
"""

from   ngams import *
import ngamsPlugInApi
import ngamsLinuxSystemPlugInApi, ngamsEscaladeUtils


def ngamsLinuxOnlinePlugIn(srvObj,
                           reqPropsObj = None):
    """
    Function mounts all NGAMS disks and loads the kernel module for the IDE
    controller card. It returns the NGAMS specific disk info dictionary.

    srvObj:        Reference to instance of the NG/AMS Server
                   class (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).
```

```
    Returns:       Disk info dictionary (dictionary).
    """
    info(4,"Entering ngamsLinuxOnlinePlugIn() ...")
    rootMtPr = srvObj.getCfg().getMountRootDirectory()
    parDic = ngamsPlugInApi.\
             parseRawPlugInPars(srvObj.getCfg().getOnlinePlugInPars())
    stat = ngamsLinuxSystemPlugInApi.insMod(parDic["module"])
    if (stat == 0):
        msg = "Kernel module " + parDic["module"] + " loaded"
        info(1, msg)

        # Old format = unfortunately some Disk IDs of WDC/Maxtor were
        # generated wrongly due to a mistake by IBM, which lead to a wrong
        # implementation of the generation of the Disk ID.
        if (not parDic.has_key("old_format")):
            raise Exception, "Missing Online Plug-In Parameter: old_format=0|1"
        else:
            oldFormat = int(parDic["old_format"])

        # The controllers Plug-In Parameter, specifies the number of controller
        # in the system.
        if (not parDic.has_key("controllers")):
            controllers = None
        else:
            controllers = parDic["controllers"]

        # Get start index for the 3ware disk devices.
        if (not parDic.has_key("dev_start_idx")):
            devStartIdx = "a"
        else:
            devStartIdx = parDic["dev_start_idx"]

        # Select between 3ware WEB Interface and 3ware Command Line Tool.
        if (parDic["uri"].find("http") != -1):
            diskDic = ngamsEscaladeUtils.parseHtmlInfo(parDic["uri"], rootMtPr)
        else:
            diskDic = ngamsEscaladeUtils.\
                      parseCmdLineInfo(rootMtPr,
                                       controllers,
                                       oldFormat,
                                       slotIds = ["*"],
                                       buf = "",
                                       devStartIdx = devStartIdx)

        ngamsLinuxSystemPlugInApi.removeFstabEntries(diskDic)
        ngamsLinuxSystemPlugInApi.ngamsMount(srvObj, diskDic,
                                             srvObj.getCfg().getSlotIds())
        info(4,"Leaving ngamsLinuxOnlinePlugIn()")
        return diskDic
    else:
        errMsg = "Problem executing ngamsLinuxOnlinePlugIn"
        errMsg = genLog("NGAMS_ER_ONLINE_PLUGIN", [errMsg])
        error(errMsg)
        raise Exception, errMsg


if __name__ == '__main__':
    """
    Main function.
    """
    import sys
    import ngamsConfig, ngamsDb

    setLogCond(0, "", 0, "", 1)

    if (len(sys.argv) != 2):
        print "\nCorrect usage is:\n"
        print "% python ngamsLinuxOnlinePlugIn <NGAMS cfg>\n"
        sys.exit(0)

    ngamsCfgObj = ngamsConfig.ngamsConfig()
    ngamsCfgObj.load(sys.argv[1])
    dbConObj = ngamsDb.ngamsDb(ngamsCfgObj.getDbServer(),
                               ngamsCfgObj.getDbName(),
                               ngamsCfgObj.getDbUser(),
                               ngamsCfgObj.getDbPassword())
    dbConObj.query("use " + ngamsCfgObj.getDbName())
    diskDic = ngamsLinuxOnlinePlugIn(dbConObj, ngamsCfgObj)
    print "Disk Dictionary = ", str(diskDic)


# EOF
```

Figure 35: Example System Online Plug-In (FILE: "ngams/ngamsPlugIns/ngamsLinuxOnlinePlugIn.py".

1.59 EXPERT: The System Offline Plug-In

The purpose of the System Offline Plug-In, is to prepare the system for the Offline State, where it should be put to its 'standby condition'. During this procedure, the disks could be unmounted and other actions performed like e.g. unloading of SW modules used for accessing the Storage Media.

.0.8 EXPERT*:* Interface of a System Offline Plug-In

The function interface of a System Offline Plug-In is the same as for the System Online Plug-In (see .0.6). A System Offline Plug-In does not return any information to NG/AMS. An exception must be thrown in case errors occur during the process of bringing the system to the Offline State.

.0.9 EXPERT*:* Example System Offline Plug-In

In the following an example System Offline Plug-In, used for the NGAS installation for WFI at the La Silla 2.2m telescope. It is perhaps not a very good example of such a plug-in since most of the code is distributed in other modules. Check the Python source files "ngams/ngamsPlugIns/ngamsEscaladeUtils.py" and "ngams/ngamsPlugIns/ngamsLinuxSystemPlugInApi.py" for further information.

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsLinuxOfflinePlugIn.py,v 1.20 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ---------------------------------------------------
# jknudstr  10/05/2001  Created.
#

"""
Module that contains a System Offline Plug-In used by the ESO NGAS
installations.
"""

from   ngams import *
import ngamsPlugInApi
import ngamsLinuxSystemPlugInApi, ngamsEscaladeUtils


def ngamsLinuxOfflinePlugIn(srvObj,
                            reqPropsObj = None):
    """
    Function unmounts all NGAMS disks and removes the kernel module for
    the IDE controller card.

    srvObj:        Reference to instance of the NG/AMS Server class
                   (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).

    Returns:       Void.
    """
    rootMtPr = srvObj.getCfg().getMountRootDirectory()
    parDicOnline = ngamsPlugInApi.\
                   parseRawPlugInPars(srvObj.getCfg().getOnlinePlugInPars())

    # Old format = unfortunately some Disk IDs of WDC/Maxtor were
    # generated wrongly due to a mistake by IBM, which lead to a wrong
    # implementation of the generation of the Disk ID.
    if (not parDicOnline.has_key("old_format")):
        raise Exception, "Missing Online Plug-In Parameter: old_format=0|1"
    else:
        oldFormat = int(parDicOnline["old_format"])

    # The controllers Plug-In Parameter, specifies the number of controller
    # in the system.
    if (not parDicOnline.has_key("controllers")):
        controllers = None
    else:
        controllers = parDicOnline["controllers"]

    # Select between 3ware WEB Interface and 3ware Command Line Tool.
    if (parDicOnline["uri"].find("http") != -1):
        diskDic = ngamsEscaladeUtils.\
                  parseHtmlInfo(parDicOnline["uri"], rootMtPr)
```

```
    else:
        diskDic = ngamsEscaladeUtils.parseCmdLineInfo(rootMtPr, controllers,
                                                      oldFormat)

    parDicOffline = ngamsPlugInApi.\
                    parseRawPlugInPars(srvObj.getCfg().getOfflinePlugInPars())

    # This is only unmounting the NGAMS disks and may lead to problems
    # if someone mounts other disks off-line.
    if (parDicOffline.has_key("unmount")):
        unmount = int(parDicOffline["unmount"])
    else:
        unmount = 1
    if (unmount):
        ngamsLinuxSystemPlugInApi.ngamsUmount(diskDic,
                                              srvObj.getCfg().getSlotIds())
        stat = ngamsLinuxSystemPlugInApi.rmMod(parDicOnline["module"])
        if (stat):
            errMsg = "Problem executing ngamsLinuxOfflinePlugIn! " +\
                     "The system is in not in a safe state!"
            errMsg = genLog("NGAMS_ER_OFFLINE_PLUGIN", [errMsg])
            error(errMsg)
            raise Exception, errMsg
        msg = "Kernel module " + parDicOnline["module"] + " unloaded"
        info(1,msg)


if __name__ == '__main__':
    """
    Main function.
    """
    import sys
    import ngamsConfig, ngamsDb

    setLogCond(0, "", 0, "", 1)

    if (len(sys.argv) != 2):
        print "\nCorrect usage is:\n"
        print "% python ngamsLinuxOfflinePlugIn <NGAMS cfg>\n"
        sys.exit(0)

    ngamsCfgObj = ngamsConfig.ngamsConfig()
    ngamsCfgObj.load(sys.argv[1])
    dbConObj = ngamsDb.ngamsDb(ngamsCfgObj.getDbServer(),
                               ngamsCfgObj.getDbName(),
                               ngamsCfgObj.getDbUser(),
                               ngamsCfgObj.getDbPassword())
    dbConObj.query("use " + ngamsCfgObj.getDbName())
    ngamsLinuxOfflinePlugIn(dbConObj, ngamsCfgObj)


# EOF
```

Figure 36: Example System Offline Plug-In (FILE: "ngams/ngamsPlugIns/ngamsLinuxOfflinePlugIn.py").

1.60 EXPERT: The Label Printer Plug-In

The purpose of the Label Printer Plug-In is to print a label on request from NG/AMS on the label printer installed on the NGAS Host. The plug-in must generate the appropriate control sequence of characters in order to request the printer to produce the label. Also other actions needed to control the printer should be taken care of by the plug-in. I.e., the plug-in could be seen as a high-level/intelligent printer driver.

.0.10 EXPERT*:* Interface of a Label Printer Plug-In

A Label Printer Plug-In must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Label Printer Plug-In has an interface as shown in ***Figure 37***.

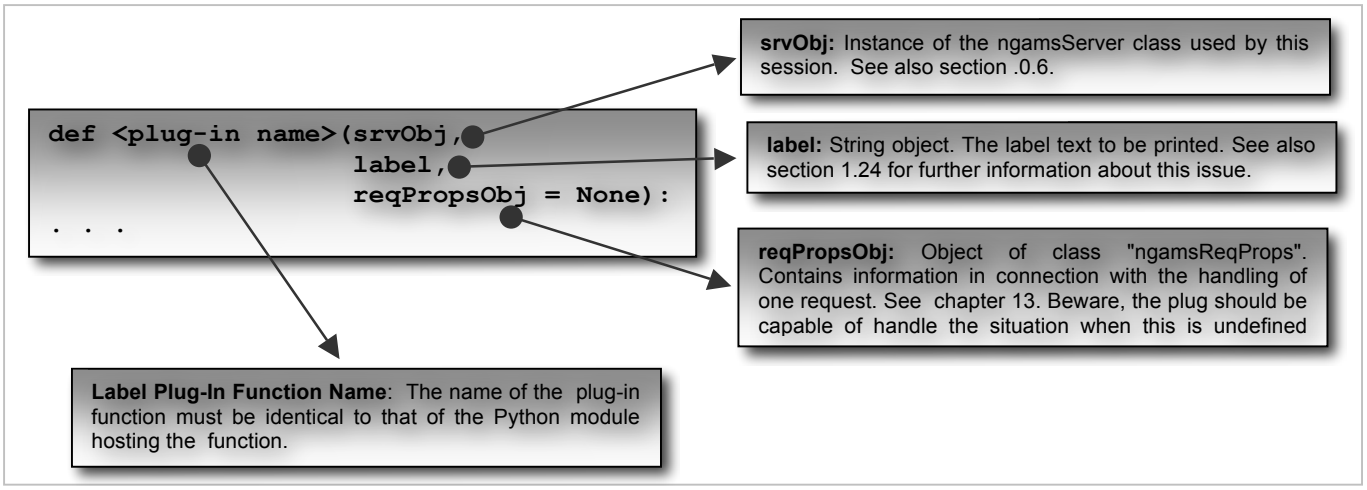

Figure 37: Function interface of a Label Printer Plug-In.

A Label Printer Plug-In does not return any data to NG/AMS. An exception must be thrown in case errors occur during the printing process.

.0.11 EXPERT: Example of a Label Printer Plug-In

In the following the source code of an example is shown. This is used to control a Brother label printer (Brother P-Touch, 9200 DX).

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsBrotherPT9200DxPlugIn.py,v 1.26 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# awicenec/
# jknudstr  10/05/2001  Created
#

"""
This module contains a plug-in driver for printing labels on
the Brother PT-9200DX label printer.
"""

import sys, time
from   ngams import *
import ngamsPlugInApi, ngamsConfig

# IMPL: Build in a semaphore protection to avoid that more than one
#       request is executed simultaneously attempting to print a label.
#       Parallel access to the label printer may harm the printer.
#       The entire code of the Label Printer Plug-In
#       (ngamsBrotherPT9200DxPlugIn) should be semaphore protected and
#       before exiting the function, a sleep of ~5s should be done. After
#       that the semaphore can be released. If another requests tries to
#       enter the semaphore protected region while one request is being
#       executed, an exception will be raised.
#       Implementing this, the _labelPrinterSem semaphore can be removed
#       from the ngamsLabelCmd.py module.

def genFontsDictionary(fnm):

    """
    Function reads the contents of a bitmap character file <fnm>.
    The character contents of this file has to be compliant with the  keys:

    keys = ['Header','-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
            ':', 'A','B','C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
            'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
            'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
            'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
            'z', 'Trailer']

    These keys are used to fill a dictionary with the bitmaps and can
    then be used to print strings on the Brother pTouch 9200DX
```

```
    printer.

    Synopsis:  charDict = ngamsGetCharDict(<fnm>)

    fnm:        Filename of font definition file (string).

    Returns:    Return value is a dictionary with the keys
                given above (dictionary).
    """
    keys = ['Header','-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
            ':', 'A','B','C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
            'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
            'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
            'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
            'z', 'Trailer']

    try:
        f = open(fnm)
        charArr = f.read()
        f.close()
    except Exception, e:
        error(str(e))
        errMsg = "Problems opening CharDict file (" + str(e) + ") "
        raise Exception, errMsg

    charArr = charArr.split('ZG')
    charDict = {}
    i = 0
    if len(charArr) != len(keys):
        errMsg = 'Wrong number of characters in CharDict file: ' + fnm
        error(str(e))
        raise Exception, errMsg

    for k in keys:
        if k == 'Header' or k == 'Trailer':
            charDict.update({k:charArr[i]})
        else:
            charDict.update({k:'G'+charArr[i]})   # put the G back
        charDict.update({' ':'ZZZZZZZZZZZZZ'})    # add a blank
        i = i + 1

    return charDict


def ngamsBrotherPT9200DxPlugIn(srvObj,
                               label,
                               reqPropsObj = None):
    """
    Driver for printing labels on the label printer Brother PT-9200DX.

    srvObj:           Reference to instance of the NG/AMS Server
                      class (ngamsServer).

    label:            Label text to print (string).

    reqPropsObj:      NG/AMS request properties object (ngamsReqProps).

    Returns:          Void.
    """
    plugInPars = srvObj.getCfg().getLabelPrinterPlugInPars()
    info(2,"Executing plug-in ngamsBrotherPT9200DxPlugIn with parameters: "+
         plugInPars + " - Label: " + label + " ...")
    parDic = ngamsPlugInApi.parseRawPlugInPars(plugInPars)

    # Get the font bit pattern dictionary.
    fontDic = genFontsDictionary(parDic["font_file"])

    # Generate the printer control code.
    printerCode = fontDic["Header"]
    for i in range(len(label)):
        if (not fontDic.has_key(label[i])):
            errMsg = "No font defintion for character: \"" + label[i] +\
                     "\" - in font definition file: " + parDic["font_file"] +\
                     " - cannot generate disk label: " + label
            error(errMsg)
            ngamsPlugInApi.notify(srvObj.getCfg(), NGAMS_NOTIF_ERROR,
                                  "ngamsBrotherPT9200DxPlugIn: " +\
                                  "ILLEGAL CHARACTER REQ. FOR PRINTING",
                                  errMsg)
            raise Exception, errMsg

        printerCode = printerCode + fontDic[label[i]]
    printerCode = printerCode + fontDic["Trailer"]
```

```
    # Generate printer file, write the printer control code.
    tmpDir = ngamsPlugInApi.getTmpDir(srvObj.getCfg())
    ngasId = ngamsPlugInApi.genNgasId(srvObj.getCfg())
    printerFilename = os.path.normpath(tmpDir + "/ngamsLabel_" + ngasId+".prn")
    fo = open(printerFilename, "w")
    fo.write(printerCode)
    fo.close()

    # Write the printer code file to the device.
    stat, out = ngamsPlugInApi.execCmd("cat " + printerFilename + " > " +\
                                       parDic["dev"])
    if (not getUnitTest()): os.system("rm -f " + printerFilename)
    if (stat != 0):
        errMsg = "Problem occurred printing label! Error: " + str(out)
        error(errMsg)
        ngamsPlugInApi.notify(srvObj.getCfg(), NGAMS_NOTIF_ERROR,
                              "ngamsBrotherPT9200DxPlugIn: " +\
                              "PROBLEM PRINTING LABEL", errMsg)
        raise Exception, errMsg

    info(2,"Executed plug-in ngamsBrotherPT9200DxPlugIn with parameters: "+
         plugInPars + " - Label: " + label + " ...")


if __name__ == '__main__':
    """
    Main function.
    """
    setLogCond(0, "", 0, "", 5)
    if (len(sys.argv) != 3):
        print "\nCorrect usage is:\n"
        print "% (python) ngamsBrotherPT9200DxPlugIn <NGAMS CFG> <text>\n"
        sys.exit(1)
    cfg = ngamsConfig.ngamsConfig()
    cfg.load(sys.argv[1])
    ngamsBrotherPT9200DxPlugIn(cfg, sys.argv[2])


# EOF
```

Figure 38: Example Label Printer Plug-In (FILE: "ngams/ngamsPlugIns/ngamsBrotherPT9200DxPlugIn.py").

1.61 EXPERT: The Data Archiving Plug-In - DAPI

The purpose of the DAPI, is to handle the archiving of data files. There are often specific aspects to take into account while archiving various kinds of data. The DAPIs make it possible to adapt NG/AMS for handling new (user specific) types of data. I.e., nothing is hard coded in the SW in connection with the data handling.

When the NG/AMS Server receives an Archive Request, a thread is spawned to handling the request. It first classifies the data and finds the appropriate Storage Set on which to store the file. Subsequently it receives the data into an intermediate file with a unique name in the Staging Area on the Main Disk of the target Storage Set. The target Storage Set is determined from the NG/AMS Configuration. From the mime-type of the data a suitable Stream is found, and afterwards a suitable Storage Set.

After having received the file, the DAPI configured for handling that type of data is invoked to carry out specific tasks to be done during the archiving.

The main tasks of a DAPI are as follows:

Task	*Description*
Data Consistency Checking	Usually it is advisable to carry out a check of the data before archiving it. Such a check could e.g. be to calculate the checksum of the file, or to check that certain parameters are properly set in the data file. If inconsistencies are found, the file should be moved to the Bad Files Directory on the Target Disk. This however, is done by NG/AMS. If a file is found to be bad, an exception should be thrown, which contains the error mnemonic "NGAMS_ER_BAD_FILE"; see the example DAPI, section .0.14 for clarification on this topic (FUNCTION: "ngams/ngamsPlugIns/ngamsFitsPlugIn.checkCheckSum()").
Data Processing	Before archiving a file, it is often necessary to do some processing. It could be something as simple as compressing the file, but in principle there are no limits to the kind of data processing that can be carried out. If the processing changes the mime-type of the file, it is important that the DAPI returns the new mime-type to NG/AMS.

Generating Final (Target) Filename	The target filename of a data file may be generated from parameters in the header. The filename is composed by the Mount Point of the disk plus the Path Prefix from the configuration. How the rest of the filename is generated is up to the plug-in implementation.
Generating Standard DAPI Return Value	A number of parameters like File ID, File Version, file size, Disk ID and more for the file archived must be returned to NG/AMS in order to update the NGAS DB accordingly. A convenience function provided in "ngamsPlugInApi" should be used for this (FUNCTION: “ngams/ngamsPlugInApi.genDapiSuccessStat()”).

After the DAPI has finished execution, NG/AMS will move the processed file to its final destination (which was decided by the DAPI). Also the NGAS DB is updated by NG/AMS with the information about the new file. If replication is requested, the file is replicated and the DB updated, also with the information for the Replication File.

The DAPI is only concerned with the Main File. If a Replication File should be produced this is entirely handled by NG/AMS. The DAPI can indicate to NG/AMS that no replication should be carried out by setting the flag “ngamsReqProps.setNoReplication(1)”. Note, that if no Replication Disk is specified in the Storage Set, no replication is performed automatically. If replication is switched off via the configuration file, by the DAPI or if no Replication Disk is specified in the configuration, the information about the Replication File is not updated in the DB.

File Versioning can be switched off by the client issuing the file for archiving using the parameter “no_versioning=1” (see 1.79). The DAPI must use the value of “ngamsReqProps.getNoVersioning()” to check if File Versioning is active. This is handled automatically by the convenience function ”ngamsPlugInApi.genFileInfo()” and the DAPI does not need to worry about this.

The diagram in ***Figure 39*** shows the actions carried out by NG/AMS and the DAPI while handling an Archive Request. Only the main actions are shown in the figure. Behind the scenes a number of other tasks are performed in order to archive a file properly.

As seen in ***Figure 39***, the handling of an Archive Request is initiated by a data provider sending an Archive Pull or an Archive Push Request to the NG/AMS Server (1). NG/AMS determines the type of data (mime-type)[8] and from this the Target Storage Set is determined. Subsequently the data is received into the Staging Area on the Target Main Disk (2). Subsequently the DAPI is invoked (3), which does the necessary data consistency checking, processing and extraction of information from the file (4). The DAPI returns control to NG/AMS and delivers back a set of information needed by NG/AMS for the further processing of the file (5). NG/AMS stores the Main File in its final location on the Main Disk (6). Then the information about the new Main File is updated in the NGAS DB (7). If replication is enabled and a Replication Disk is defined, NG/AMS creates the Replication File (8). Afterwards the information for the Replication File is updated in the DB (9). NG/AMS can either return an immediate reply to the client issuing the Archive Request or it can return a reply when the file has been successfully (or unsuccessfully) handled.

[8] If the mime-type is not specified explicitly in the Archive Request, NG/AMS will attempt to determine the mime-type from the extension specified in the URI of the data file issued for archiving. In this case the value of the mime-type should be set to the generic mime-type “ngas/archive-request” to signal to NG/AMS to figure out the mime-type.

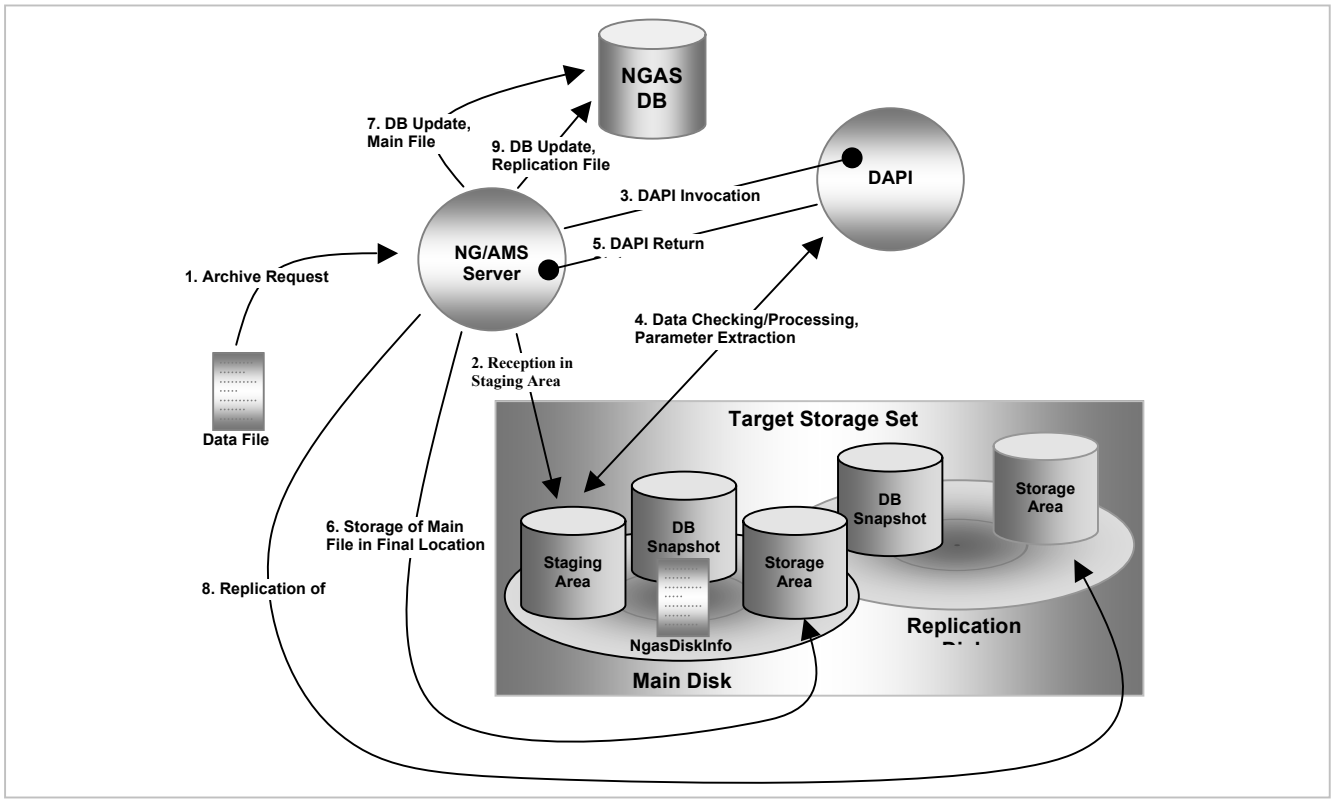

Figure 39: Handling of an Archive Request.

Note, that the DAPI is a function running within the same Python interpreter as the NG/AMS Server process.

.0.12 EXPERT*:* Interface of a DAPI

The DAPI must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual DAPI, which is invoked by NG/AMS.

A DAPI has an interface as shown in ***Figure 40***.

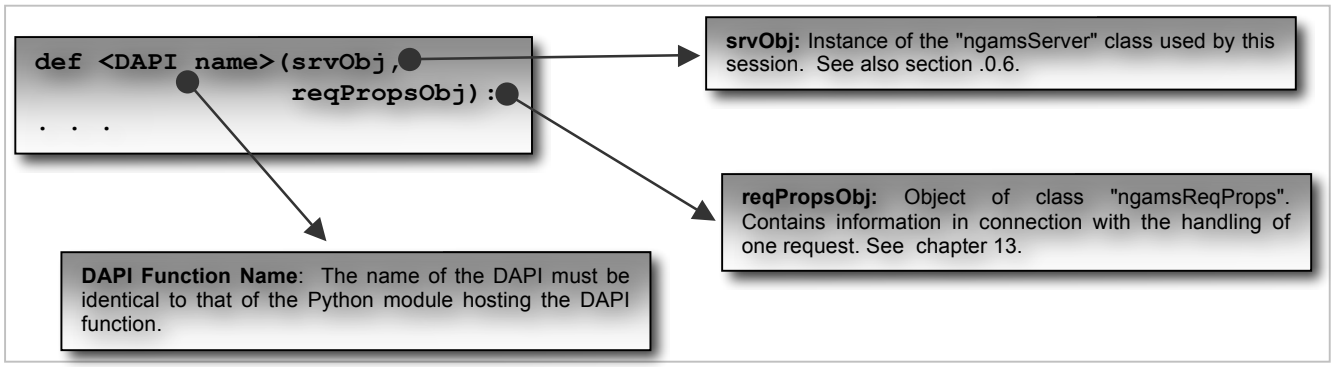

Figure 40: Function interface of a DAPI.

A DAPI must perform the following return when finishing execution:

```
return ngamsPlugInApi.genDapiSuccessStat(diskId,
                                         relFilename,
                                         fileId,
                                         fileVersion,
                                         format,
                                         fileSize,
                                         uncomprSize,
                                         compression,
                                         relPath,
```

```
                                    slotId,
                                    fileExists,
                                    completeFilename)
```

Figure 41: DAPI return statement.

The return parameters of a DAPI are as follows:

Parameter	*Type*	*Description*
diskId	String	Disk ID of file.
relFilename	String	Filename relative to mount point.
fileId	String	File ID allocated to the file by the DAPI.
fileVersion	Integer	Version of the file.
format	String	Format (or mime-type) of the file. Only mime-types defined in the NG/AMS Configuration are accepted.
fileSize	Integer	Size of the file as it is archived.
uncomprSize	Integer	Uncompressed size of the file. I.e., if the file was compressed, this is the original size before archiving/compression.
compression	String	Compression method used to compress file. Should be the command invoked to compress the file, e.g. "compress".
relPath	String	Path relative to the mount point of the target disk.
slotId	String	Slot ID of slot in which the Main Disk is installed.
fileExists	Integer	Indicates if the file already existed on the target disk. In case yes, this should be 1, otherwise 0.
completeFilename	String	The complete name of the file as it should be. The complete name must be generated by the DAPI.

Table 15: Return parameters of a DAPI.

In case of an error, a DAPI must raise an exception using one of the following NG/AMS Log Codes defined in the NG/AMS Log Definition (chapter 9715):

Log Code	*Description*
NGAMS_ER_DAPI	A error was encountered while executing the plug-in. This is not related to the data file itself. If Back-Log Buffering is enabled, NG/AMS will Back-Log Buffer the file and will try to archive it later.
NGAMS_ER_DAPI_BAD_FILE	The file was classified as bad by the plug-in. The staging file will be moved to the Bad Files Directory by NG/AMS.
NGAMS_ER_DAPI_RM	For some reason the plug-in decided to discard the file. The file will be removed by NG/AMS.

Table 16: Exception that can be raised by a DAPI.

The plug-in should raise the exception, by building up a proper log message using the NG/AMS function “genLog()”, e.g.:

```
def getComprExt(comprMethod):
    """
    Determine the extension for the given type of compression specified.

    comprMethod:   Compression method e.g. 'compress' or 'gzip' (string).

    Returns:       Extension used by the given compression method (string).
    """
    if (comprMethod.find("compress") != -1):
        return "Z"
    elif (comprMethod.find("gzip") != -1):
        return "gz"
    else:
        errMsg = "Unknown compression method specified: " + comprMethod
        errMsg = genLog("NGAMS_ER_DAPI", [errMsg])
        raise Exception, errMsg
```

The contents of the error message to provide with the NG/AMS Log Code can be generated freely by the plug-in to describe the type of problem encountered. NG/AMS only probes for the error code and does not analyze the error message as such.

It is important that the plug-in catches exception that might be thrown in methods/functions called by the plug-in. The error message of the exception from the method invoked can be used as error message for the NG/AMS Log Code.

.0.13 EXPERT: Overall Structure & Algorithm of a DAPI

The overall structure of a DAPI Python source and in particular the DAPI function itself is shown in ***Figure 42***.

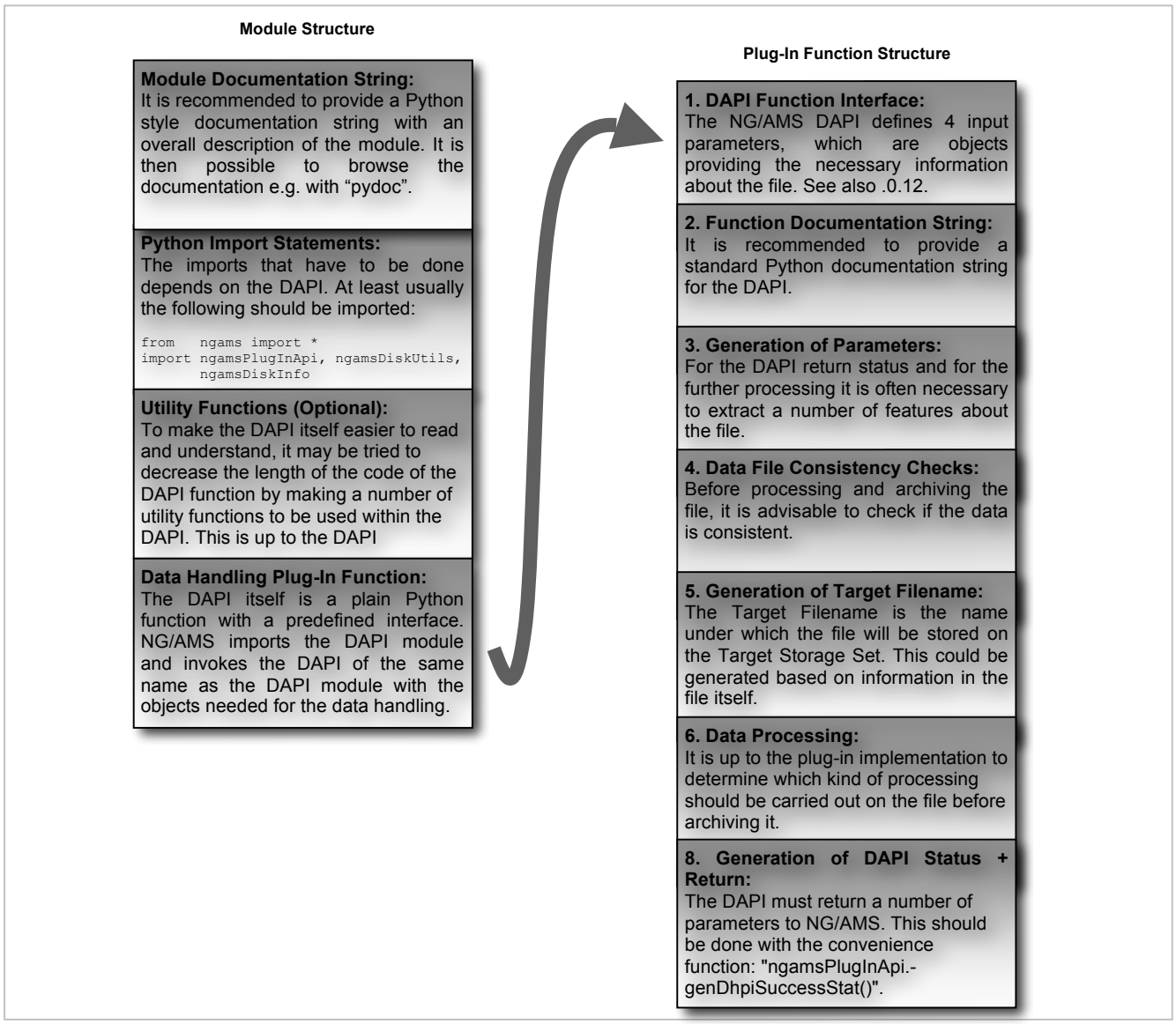

Figure 42: Typical structure of a DAPI module and a DAPI function.

The exact sequence of the actions performed and the actions themselves, may vary from DAPI to DAPI. I.e., maybe the data processing is done before the generation of the final target filename. In section .0.14 an example DAPI module is shown.

.0.14 EXPERT: Example DAPI - WFI/FITS File DAPI

In the following an example DAPI, which is used for archiving FITS files at the 2.2m telescope at La Silla is shown:

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsFitsPlugIn.py,v 1.61 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------
# jknudstr  10/05/2001  Created
#

"""
This Data Archiving Plug-In is used to handle reception and processing
of FITS files.

Note, that the plug-in is implemented for the usage at ESO. If used in other
contexts, a dedicated plug-in matching the individual context should be
implemented and NG/AMS configured to use it.
"""

import os, string
import PccUtTime
from   ngams import *
import ngamsPlugInApi, ngamsDiskUtils, ngamsDiskInfo


def getComprExt(comprMethod):
    """
    Determine the extension for the given type of compression specified.

    comprMethod:   Compression method e.g. 'compress' or 'gzip' (string).

    Returns:       Extension used by the given compression method (string).
    """
    if (comprMethod.find("compress") != -1):
        return "Z"
    elif (comprMethod.find("gzip") != -1):
        return "gz"
    else:
        errMsg = "Unknown compression method specified: " + comprMethod
        errMsg = genLog("NGAMS_ER_DAPI", [errMsg])
        raise Exception, errMsg


def getDpIdInfo(filename):
    """
    Generate the File ID (here DP ID) for the file.

    filename:   Name of FITS file (string).

    Returns:    Tuple containing the value of ARCFILE, the DP ID
                of the file, and the JD date. The two latter deducted from
                the ARCFILE keyword (tuple).
    """
    try:
        keyDic  = ngamsPlugInApi.getFitsKeys(filename, ["ARCFILE"])
        arcFile = keyDic["ARCFILE"][0]
        els     = string.split(arcFile, ".")
        dpId    = els[0] + "." + els[1] + "." + els[2]
        date    = string.split(els[1], "T")[0]
        # Make sure that the files are stored according to JD
        # (one night is 12am -> 12am).
        isoTime = els[1]
        ts1 = PccUtTime.TimeStamp(isoTime)
        ts2 = PccUtTime.TimeStamp(ts1.getMjd() - 0.5)
        dateDirName = string.split(ts2.getTimeStamp(), "T")[0]

        return [arcFile, dpId, dateDirName]
    except:
        err = "Did not find keyword ARCFILE in FITS file or ARCFILE illegal"
        errMsg = genLog("NGAMS_ER_DAPI_BAD_FILE", [os.path.basename(filename),
                                                   "ngamsFitsPlugIn", err])
        raise Exception, errMsg


def checkFitsFileSize(filename):
    """
    Check if the size of the FITS file is a multiple of 2880. If this
    is not the case, we through an exception.

    filename:   FITS file to check (string).

    Returns:    Void.
    """
```

```
    if (string.split(filename, ".")[-1] == "fits"):
        size = ngamsPlugInApi.getFileSize(filename)
        testVal = (size / 2880.0)
        if (testVal != int(testVal)):
            errMsg = "The size of the FITS file issued " +\
                     "is not a multiple of 2880! Rejecting file!"
            errMsg = genLog("NGAMS_ER_DAPI_BAD_FILE",
                            [os.path.basename(filename),
                             "ngamsFitsPlugIn", errMsg])
            raise Exception, errMsg


def checkChecksum(parDic,
                  filename):
    """
    Check that the checksum of the file is correct.

    parDic:    Dictionary with disk information (ngamsPhysDiskInfo objects)
               (dictionary).

    filename:  Name of FITS file (string).

    Returns:   Void.
    """
    # Only do check if the checksum_util parameter is set.
    if (not parDic.has_key("checksum_util")): return

    # Execute the checksum routine and evaluate result.
    info(2,"Invoking checksum test utility: " + parDic["checksum_util"] +\
         " on file: " + filename)
    res = ngamsPlugInApi.execCmd(parDic["checksum_util"] + " " + filename)
    if (int(res[0]) != 0):
        errMsg = "Problem occurred invoking checksum check utility: " +\
                 parDic["checksum_util"]
        errMsg = genLog("NGAMS_ER_DAPI", [errMsg])
        error(errMsg)
        raise Exception, errMsg
    if (res[1] != parDic["checksum_result"]):
        errMsg = "Executing checksum utility: " + parDic["checksum_util"] +\
                 " gave unexpected result. Result: [" + res[1] + "]. " +\
                 "Expected Result: [" + parDic["checksum_result"] + "]."
        errMsg = genLog("NGAMS_ER_DAPI_BAD_FILE", [filename, "ngamsFitsPlugIn",
                                                   errMsg])
        error(errMsg)
        raise Exception, errMsg


def prepFile(reqPropsObj,
             parDic):
    """
    Prepare the file. If it is compressed, decompress it into a temporary
    filename.

    reqPropsObj:  NG/AMS request properties object (ngamsReqProps).

    parDic:       Dictionary with parameters for the DAPI. This is generated
                  with ngamsPlugInApi.parseDapiPlugInPars() (Dictionary).

    Returns:      Tuple containing:

                    (<DP ID>, <Date Obs. Night>, <Compr. Ext.>)   (tuple).
    """
    info(4,"ngamsFitsPlugIn: Entering prepFile() ...")

    # If the file is already compressed, we have to decompress it.
    tmpFn = reqPropsObj.getStagingFilename()
    if ((tmpFn.find(".Z") != -1) or (tmpFn.find(".gz") != -1)):
        ngamsPlugInApi.execCmd("gunzip " + tmpFn)
        reqPropsObj.setStagingFilename(os.path.splitext(tmpFn)[0])
    checkFitsFileSize(reqPropsObj.getStagingFilename())
    checkChecksum(parDic, reqPropsObj.getStagingFilename())
    if (parDic.has_key("compression")):
        comprExt = getComprExt(parDic["compression"])
    else:
        comprExt = ""
    dpIdInfo = getDpIdInfo(reqPropsObj.getStagingFilename())

    info(4,"ngamsFitsPlugIn: Leaving prepFile()")
    return dpIdInfo[1], dpIdInfo[2], comprExt


def compress(reqPropsObj,
             parDic):
    """
```

```
    Compress the file if required.

    reqPropsObj:  NG/AMS request properties object (ngamsReqProps).

    parDic:       Dictionary with parameters for the DAPI. This is generated
                  with ngamsPlugInApi.parseDapiPlugInPars() (Dictionary).

    Returns:      Tupe containing uncompressed filesize, archived filesize
                  and the format (mime-type) of the resulting data file
                  (tuple).
    """
    stFn = reqPropsObj.getStagingFilename()

    # If a compression application is specified, apply this.
    uncomprSize = ngamsPlugInApi.getFileSize(stFn)
    if (parDic["compression"] != ""):
        info(2,"Compressing file using: " + parDic["compression"] + " ...")
        exitCode, stdOut =\
                  ngamsPlugInApi.execCmd(parDic["compression"] + " " + stFn)
        if (exitCode != 0):
            errMsg = "ngamsFitsPlugIn: Problems during archiving! " +\
                     "Compressing the file failed"
            raise Exception, errMsg

        stFn = stFn + "." + getComprExt(parDic["compression"])
        # Remember to update Staging Filename in the Request Properties Object.
        reqPropsObj.setStagingFilename(stFn)
        if (parDic["compression"].find("compress") != -1):
            format = "application/x-cfits"
        else:
            format = "application/x-gfits"
        info(2,"File compressed")
    else:
        format = reqPropsObj.getMimeType()

    archFileSize = ngamsPlugInApi.getFileSize(reqPropsObj.getStagingFilename())
    return uncomprSize, archFileSize, format


# DAPI function.
def ngamsFitsPlugIn(srvObj,
                    reqPropsObj):
    """
    Data Archiving Plug-In to handle archiving of FITS files.

    srvObj:       Reference to NG/AMS Server Object (ngamsServer).

    reqPropsObj:  NG/AMS request properties object (ngamsReqProps).

    Returns:      Standard NG/AMS Data Archiving Plug-In Status as generated
                  by: ngamsPlugInApi.genDapiSuccessStat() (ngamsDapiStatus).
    """
    info(1,"Plug-In handling data for file with URI: " +
         os.path.basename(reqPropsObj.getFileUri()))
    diskInfo = reqPropsObj.getTargDiskInfo()
    parDic = ngamsPlugInApi.parseDapiPlugInPars(srvObj.getCfg(),
                                                reqPropsObj.getMimeType())

    # Check file (size + checksum) + extract information.
    dpId, dateDirName, comprExt = prepFile(reqPropsObj, parDic)

    # Get various information about the file being handled.
    dpIdInfo = getDpIdInfo(reqPropsObj.getStagingFilename())
    dpId = dpIdInfo[1]
    dateDirName = dpIdInfo[2]
    fileVersion, relPath, relFilename,\
                 complFilename, fileExists =\
                 ngamsPlugInApi.genFileInfo(srvObj.getDb(), srvObj.getCfg(),
                                            reqPropsObj, diskInfo,
                                            reqPropsObj.getStagingFilename(),
                                            dpId, dpId, [dateDirName],
                                            [comprExt])

    # If a compression application is specified, apply this.
    uncomprSize, archFileSize, format = compress(reqPropsObj, parDic)

    # Generate status + return.
    info(3,"DAPI finished processing of file - returning to main application")
    return ngamsPlugInApi.genDapiSuccessStat(diskInfo.getDiskId(), relFilename,
                                             dpId, fileVersion, format,
                                             archFileSize, uncomprSize,
                                             parDic["compression"], relPath,
                                             diskInfo.getSlotId(), fileExists,
                                             complFilename)
```

```
# EOF
```

Figure 43: Example Data Archiving Plug-In (FILE: "ngams/ngamsPlugIns/ngamsFitsPlugIn.py").

1.62 EXPERT: The Register Plug-In

The Register Plug-In is used when executing a REGISTER Command (see section 1.92), to handle the processing and extraction of information from a data file, which is being registered. The plug-in is *very* similar to the Data Archiving Plug-In (chapter 1.61), but due to a few, minor differences, it has been chosen to define explicitly a new type of plug-in for the purpose of registering files. The main difference between registering and archiving of files, is that when registering files, the files stay in the location where they are, and it is not necessary to create a new target filename for these, and to move these around.

A Register Plug-In must be defined for each type of data that is going to be registered using the REGISTER Command. See also section 1.46/"Register" Element.

.0.15 EXPERT*:* Interface of a Register Plug-In

The interface of a Register Plug-In is identical to that of the Data Archiving Plug-In; see section .0.12 for further information.

.0.16 EXPERT*:* Example Register Plug-In

In the following an example Register Plug-In, which is used for archiving FITS files is shown:

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsFitsRegPlugIn.py,v 1.9 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  10/05/2001  Created
#

"""
This Data Register Plug-In is used to handle the registration of FITS files
already stored on an 'NGAS disk', which just need to be registered in the DB.

Note, that the plug-in is implemented for the usage at ESO. If used in other
contexts, a dedicated plug-in matching the individual context should be
implemented and NG/AMS configured to use it.
"""

import os, string
from   ngams import *
import ngamsPlugInApi, ngamsDiskUtils, ngamsDiskInfo, ngamsFitsPlugIn


# Data Registration Function.
def ngamsFitsRegPlugIn(srvObj,
                       reqPropsObj):
    """
    Data Registration Plug-In to handle registration of FITS files.

    srvObj:       Reference to NG/AMS Server Object (ngamsServer).

    reqPropsObj:  NG/AMS request properties object (ngamsReqProps).

    Returns:      Standard NG/AMS Data Archiving Plug-In Status as generated
                  by: ngamsPlugInApi.genDapiSuccessStat() (ngamsDapiStatus).
    """
    info(1,"Plug-In registering file with URI: " + reqPropsObj.getFileUri())
    diskInfo = reqPropsObj.getTargDiskInfo()
    parDic = ngamsPlugInApi.parseRegPlugInPars(srvObj.getCfg(),
                                               reqPropsObj.getMimeType())
    stageFile = reqPropsObj.getStagingFilename()

    # If the file is already compressed, we have to decompress it.
    procDir = ""
    if ((stageFile.find(".Z") != -1) or (stageFile.find(".gz") != -1)):
        workingFile, procDir = ngamsPlugInApi.prepProcFile(srvObj.getCfg(),
                                                           stageFile)
```

```
        ngamsPlugInApi.execCmd("gunzip " + workingFile)
        if (workingFile.find(".Z") != -1):
            workingFile = workingFile[:-2]
        else:
            workingFile = workingFile[:-3]
    else:
        workingFile = stageFile

    # Check file (size + checksum).
    ngamsFitsPlugIn.checkFitsFileSize(workingFile)
    ngamsFitsPlugIn.checkChecksum(parDic, workingFile)

    # Get various information about the file being handled.
    arcFile, dpId, dateDirName = ngamsFitsPlugIn.getDpIdInfo(workingFile)
    fileVersion, relPath, relFilename,\
                 complFilename, fileExists =\
                 ngamsPlugInApi.genFileInfoReg(srvObj.getDb(), srvObj.getCfg(),
                                               reqPropsObj, diskInfo,
                                               stageFile, dpId)

    # Generate status.
    info(4,"Generating status ...")
    fileSize = ngamsPlugInApi.getFileSize(stageFile)
    if (stageFile.find(".Z") != -1):
        format = "application/x-cfits"
        compresion = "compress"
    elif (stageFile.find(".gz") != -1):
        format = "application/x-gfits"
        compresion = "gzip"
    else:
        format = "image/x-fits"
        compresion = ""
    uncomprSize = ngamsPlugInApi.getFileSize(workingFile)

    # Delete the processing directory (would be done later by the
    # Janitor Thread, but it is better to clean up explicitly).
    if (procDir): rmFile(procDir)

    info(3,"Register Plug-In finished processing of file")
    return ngamsPlugInApi.genRegPiSuccessStat(diskInfo.getDiskId(),relFilename,
                                              dpId, fileVersion, format,
                                              fileSize, uncomprSize,compresion,
                                              relPath, diskInfo.getSlotId(),
                                              fileExists, complFilename)


# EOF
```

Figure 44: Example Register Plug-In (FILE: "ngams/ngamsPlugIns/ngamsFitsRegPlugIn.py").

1.63 EXPERT: The Data Processing Plug-In - DPPI

The purpose of the Data Processing Plug-In (DPPI) is to provide a specific type of processing to be applied on a specific type of data when data is being retrieved from an NGAS Host. Processing could be as trivial as simply uncompressing a data file, which is stored in compressed format. It could also be far more complex and involve advanced image processing and parameter extraction. How the DPPI actually processes the data, is left up to the DPPI implementation. The DPPI only has to obey the set of rules for interfacing as for any other plug-in defined for NG/AMS.

.0.17 EXPERT*:* Interface of a DPPI

The DPPI must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual DPPI, which is invoked by NG/AMS.

A DPPI has an interface as shown in ***Figure 45***.

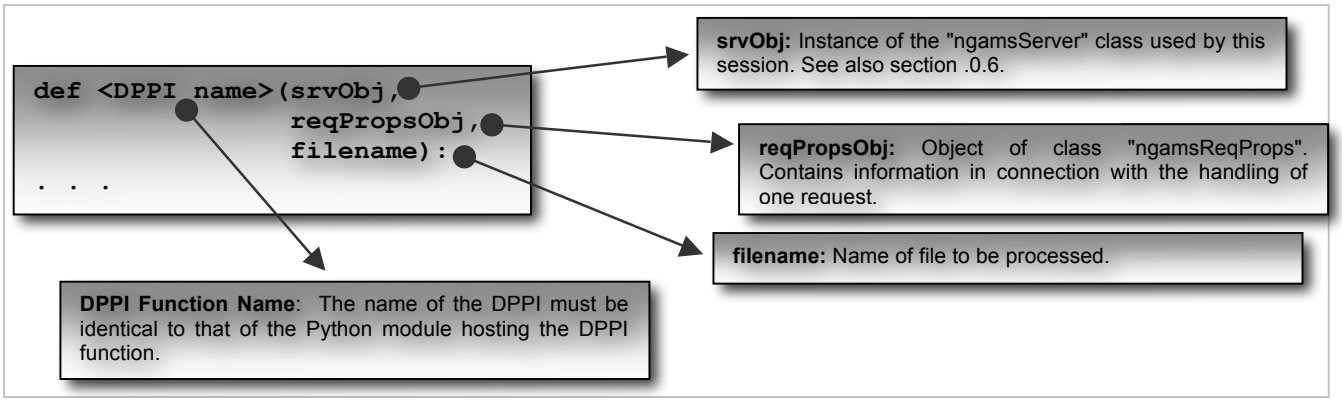

Figure 45: Function interface of a DPPI.

A DPPI must return an object of the type "ngamsDppiStatus". This again contains one or more objects of the type "ngamsDppiResult", which each refer to result data or contains the result of the processing. This means that it is possible to produce several results in a DPPI, and to have these send back to the requestor[9]. The concept of the DPPI return object is shown in ***Figure 46***.

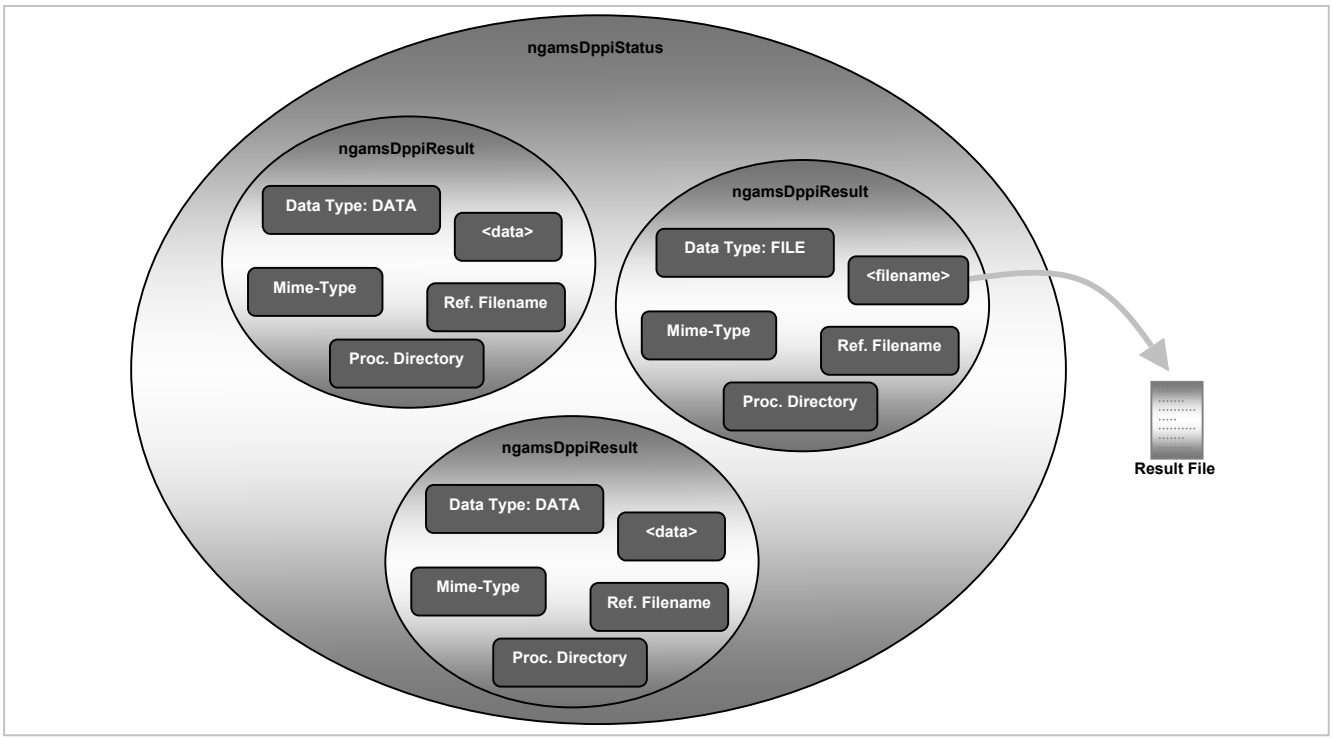

Figure 46: DPPI – structure of return data.

As shown in ***Figure 46***, the "ngamsDppiStatus" object can contain an arbitrary number of "ngamsDppiResult" objects, each containing the information of one sub-result. As can be seen in the figure, the data of a sub result can either be contained directly in the ngamsDppiResult object, or the data can be stored in an external file, which is referred to by the object. Whether to use the one or the other depends on the nature of the data. If the result consists of a smaller amount of non-binary data it is more convenient to store the data internally to avoid having to create, access and delete the result files. For larger amounts of result data and for binary data, it is recommended to use an external result file. See chapter 13 for more information about these classes.

External, temporary files (Result Files) will be deleted automatically by NG/AMS after the result data has been returned to the requestor.

[9] For actually supporting completely latter, NG/AMS needs to be extended to be able to return replies making use of the "multipart/mixed" mime-type as known from e.g. emails. This is foreseen to be supported soon.

.0.18 EXPERT: Example DPPI

In the following a very trivial example of a DPPI is shown. It is used to extract the header information of a FITS file.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsExtractFitsHdrDppi.py,v 1.16 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------
# awicenec  26/09/2002  Created
# awicenec  31/03/2004  Support for extraction of certain headers
#

"""
Contains a DDPI which is used to extract the main header from FITS files.
"""

from ngams import *
import ngamsPlugInApi, ngamsDppiStatus
import printhead


def ngamsExtractFitsHdrDppi(srvObj,
                            reqPropsObj,
                            filename):
    """
    This DPPI extracts the main header from a FITS file
    requested from the ESO Archive.

    srvObj:        Reference to instance of the NG/AMS Server
                   class (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).

    filename:      Name of file to process (string).

    Returns:       DPPI return status object (ngamsDppiStatus).

    Side effect:   This DPPI works directly on the archived file, since it is
                   read-only access.

    SPECIFIC DOCUMENTATION:

        This DPPI controls the call to the printhead module. If the

         Example URL (single line):

         http://ngasdev1:7777/RETRIEVE
                 ?file_id=MIDI.2004-02-11T04:16:04.528&
                 processing=ngamsExtractFitsHdrDppi&
                 processing_pars='header=99'

         The header parameter is optional, if not specified the primary header
         is returned.
         Valid values for the header parameter are numbers between 1 and 99 and
         the strings 'vo' or 'xf'. If numbers are specified which are either
         outside the range or bigger than the number of headers (including the
         primary) the primary header will be returned. Headers are counted from
         1 starting with the primary header. 99 is a special value as it returns
         all headers concatenated in a single file.

         If 'vo' is specified all headers are returned using a slightly modified
         VOTable (XML) format. If 'xf' is specified all headers are returned
         using the XFits (XML) format.
    """
    statusObj = ngamsDppiStatus.ngamsDppiStatus()

    pH = printhead.FitsHead(filename,struct=1)

    hind = 1 # first header only by default
    head = ''
    if (reqPropsObj.hasHttpPar("processing_pars")):
         pars = ngamsPlugInApi.parseRawPlugInPars(\
                  reqPropsObj.getHttpPar("processing_pars"))
         if not pars.has_key('header'):
             ext = '.hdr'
             hind = 1    # first header only by default
         else:           # if processing_par 'header' exists check its contents
             phead = pars['header']
             if phead.lower() in ['vo','xf']:
```

```
                ext = '.xml'
                hind = -1
                pH.parseFitsHead()
                if phead.lower() == 'vo':
                    pH.voImageHead(outfile='')
                    head = '\n'.join(pH.VoImageHead)
                else:
                    pH.xmlHead(outfile='')
                    head = '\n'.join(pH.XmlHead)
            else:
                ext = '.hdr'
                try:
                   hind = int(pars['header'])
                    if hind <= 0: hind = 1
                except:
                    hind = 1
        hind = hind - 1  # index starts at 0

    if hind == 98:
        head = ''.join(pH.HEAD)
    elif hind >= 0 and hind < len(pH.HEAD):
        head = pH.HEAD[hind]
    elif hind > len(pH.HEAD):
        head = pH.HEAD[0]

    pos = filename.rfind('.fits')
    file_id = filename[:pos]

    resFilename = file_id + ext
    try:
        mimeType = ngamsPlugInApi.determineMimeType(srvObj.getCfg(),\
                   resFilename)
    except:
        if ext == '.xml':
            mimeType = 'text/xml'
        else:
            mimeType = 'text/ascii'

    resObj = ngamsDppiStatus.ngamsDppiResult(NGAMS_PROC_DATA, mimeType,
                                             head, resFilename, '')
    statusObj.addResult(resObj)

    return statusObj


# EOF
```

Figure 47: Example Data Processing Plug-In (FILE: "ngams/ngamsPlugIns/ngamsExtractFitsHdrDppi.py").

Another example of a trivial DPPI is shown in ***Figure 48***. This DPPI is used to decompress files, which have been archived in compressed format.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsEsoArchDppi.py,v 1.12 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  08/01/2002  Created
#

"""
Contains a DDPI which is used by the ESO Archive Facility to perform the
processing in connection with a standard data request handling.
"""

from ngams import *
import ngamsPlugInApi, ngamsDppiStatus


def ngamsEsoArchDppi(srvObj,
                     reqPropsObj,
                     filename):
    """
    This DPPI performs the processing neccessary for the files
    requested from the ESO Archive (by the Data Requestor).

    srvObj:        Reference to instance of the NG/AMS Server
                   class (ngamsServer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).
```

```
    filename:      Name of file to process (string).

    Returns:       DPPI return status object (ngamsDppiStatus).
    """
    statusObj = ngamsDppiStatus.ngamsDppiStatus()

    # Decompress the file if the last extension is "Z".
    if (filename.split(".")[-1] == "Z"):
        procFilename, procDir = ngamsPlugInApi.prepProcFile(srvObj.getCfg(),
                                                            filename)
        exitCode, stdOut, stdErr = ngamsPlugInApi.\
                                   execCmd("uncompress " + procFilename)
        if (exitCode != 0):
            errMsg = "ngamsEsoArchDppi: Problems during archiving! " +\
                     "Decompressing the file: " + filename + " failed. " +\
                     "Error message: " + str(stdErr)
            raise Exception, errMsg
        resFilename = procFilename[0:-2]
    else:
        resFilename = filename
        procDir = ""
    mimeType = ngamsPlugInApi.determineMimeType(srvObj.getCfg(), resFilename)
    resObj = ngamsDppiStatus.ngamsDppiResult(NGAMS_PROC_FILE, mimeType,
                                             resFilename, resFilename, procDir)
    statusObj.addResult(resObj)

    return statusObj


# EOF
```

Figure 48: Example Data Processing Plug-In (FILE: "ngams/ngamsPlugIns/ngamsEsoArchDppi.py").

1.64 EXPERT: The Data Checksum Plug-In

The Data Checksum Plug-In is a simple plug-in used to generate the checksum value for a data file being archived. This value is written in the record for the file in the NGAS DB, and used later on to check periodically if the file is in a 'good condition'. I.e., that it is not damaged or corrupted in any way. The Data Checksum Plug-In is invoked by NG/AMS after the DAPI has finished the data type specific processing.

.0.19 EXPERT*:* Interface of a Data Checksum Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Data Checksum Plug-In has an interface as shown in ***Figure 49***.

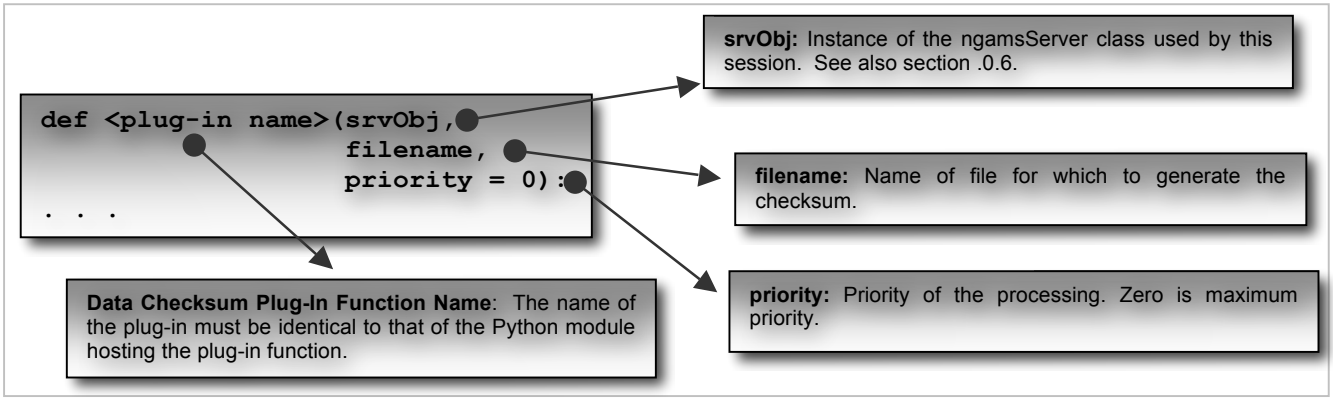

Figure 49: Function interface of a Data Checksum Plug-In (DCPI).

A Data Checksum Plug-In must return the calculated checksum value as a string.

.0.20 EXPERT*:* Example Data Checksum Plug-In

In the following the source code of a small example Data Checksum Plug-In is shown. It generates the checksum based on routines built-into Python.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsGenCrc32.py,v 1.20 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  23/01/2002  Created
#

"""
Checksum Plug-In to generate the checksum stored in the ngas_files tables
in connection with each file archived into NGAS.
"""

import sys, time

import binascii
from   ngams import *


def ngamsGenCrc32(srvObj,
                  filename,
                  priority = 0):
    """
    Plug-in to generate CRC-32 checksum for an archived data file.

    srvObj:       Reference to instance of NG/AMS Server class (ngamsServer).

    filename:     Name of file to generate checksum for (string).

    priority:     Is used by NG/AMS to make the plug-in consume less
                  CPU. A value of 0 means highest priority (integer/[0; oo]).

    Returns:      CRC-32 checksum for file (string).
    """
    fo = open(filename, "r")
    buf = fo.read(524288)
    crc = 0
    while (buf != ""):
        crc = binascii.crc32(buf, crc)
        if (priority): time.sleep(priority * 0.001)
        buf = fo.read(524288)
    fo.close()
    return str(crc)


if __name__ == '__main__':
    """
    Main routine to calculate checksum for a file given as input parameter.
    """
    if (len(sys.argv) != 2):
        print "\nCorrect usage is:\n"
        print "% ngamsGenCrc32.py <filename> [<priority [0..oo]>]\n"
        sys.exit(1)
    filename = sys.argv[1]
    if (len(sys.argv) == 3):
        priority = int(sys.argv[2])
    else:
        priority = 0
    checksum = ngamsGenCrc32(None, filename, priority)
    sys.stdout.write(checksum)


# EOF
```

Figure 50: Example Data Checksum Plug-In (FILE: "ngams/ngamsPlugIns/ngamsGenCrc32.py").

1.65 EXPERT: The Suspension Plug-In

When an NG/AMS Server is suspending itself after the specified suspension time-out has elapsed (CFG: "NgamsCfg.HostSuspension:IdleSuspensionTime"), it invokes the specified Suspension Plug-In (CFG: "NgamsCfg.HostSuspension:SuspensionPlugIn"), which actually carries out the actions needed to suspend the host (see also 1.30). In the simplest case the Suspension Plug-In might simply invoke a "shutdown" command (on UNIX) as root to shut down the host, but in principle there are no restrictions to which kind of actions that are performed.

.0.21 EXPERT*:* Interface of a Suspension Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Suspension Plug-In has an interface as shown in ***Figure 51***.

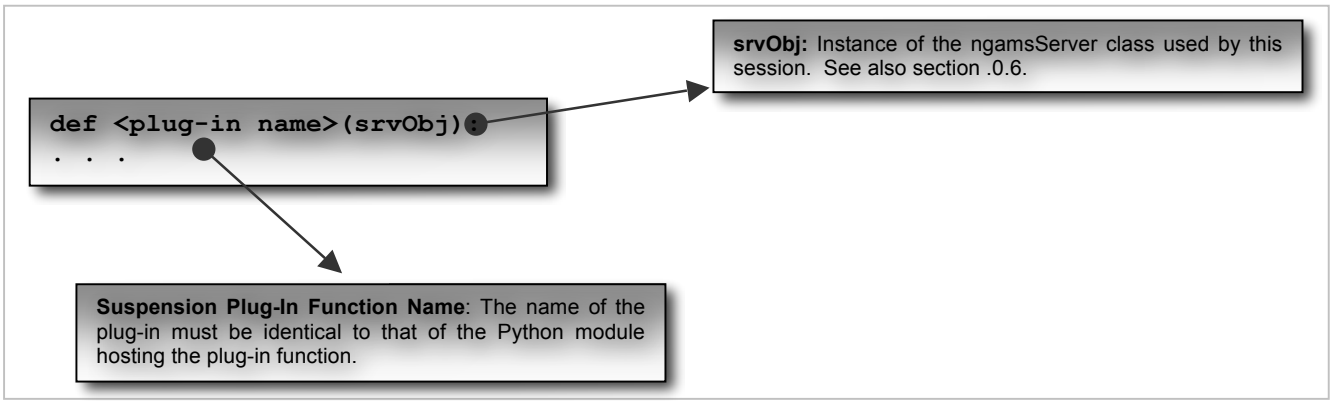

Figure 51: Function interface of a Suspension Plug-In.

A Suspension Plug-In does not return control to the NG/AMS Server but will in fact terminate this after having done various clean-up that might be necessary.

.0.22 EXPERT*:* Example Suspension Plug-In

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsSuspensionPlugIn.py,v 1.7 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  23/01/2002  Created
#

"""
Test Suspension Plug-In to simulate the NGAS host suspension.
"""

import commands

from   ngams import *
import ngamsDiskInfo


def ngamsSuspensionPlugIn(srvObj):
    """
    Suspension Plug-In to suspend an NGAS host.

    srvObj:         Reference to instance of the NG/AMS Server (ngamsServer).

    Returns:        Void.
    """
    info(4,"Suspension Plug-In executing ...")
    commands.getstatusoutput("sudo /sbin/shutdown -h now")
    info(4,"Leaving Suspension Plug-In")


# EOF
```

Figure 52: Example Suspension Plug-In (FILE: "/ngams/ngamsPlugIns/ngamsSuspensionPlugIn.py").

1.66 EXPERT: The Wake-Up Plug-In

The Wake-Up Plug-In is used by an NG/AMS Server that has been requested to wake-up an NGAS Host that has suspended itself. The Suspension/Wake-Up Service is described in section 1.30. The actions to be carried out, depends on the HW and on the system configuration. Usually a Wake-Up Plug-In will send a message to a device connected to the network to indicate it to start up an NGAS Host; this device could e.g. be the network card of the host.

.0.23 EXPERT: Interface of a Wake-Up Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Wake-Up Plug-In has an interface as shown in ***Figure 53***.

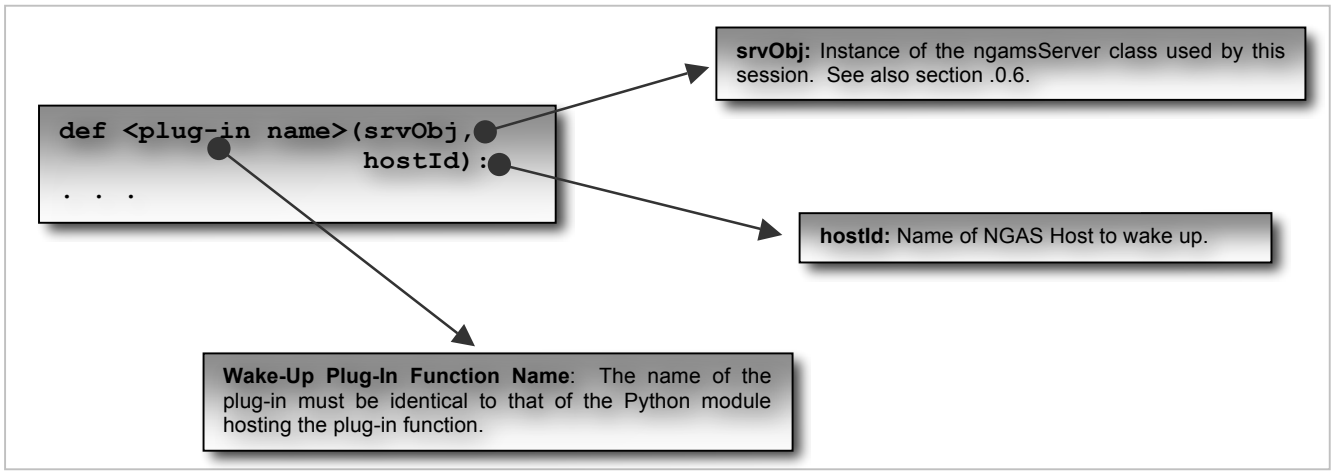

Figure 53: Function interface of a Wake-Up Plug-In.

A Wake-Up Plug-In does not return any value to the NG/AMS Server after execution.

.0.24 *EXPERT:* Example Wake-Up Plug-In

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsWakeUpPlugIn.py,v 1.6 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  23/01/2002  Created
#

"""
Test Wakeup Plug-In to simulate the NGAS host suspension.
"""

import commands

from   ngams import *
import ngamsHighLevelLib


def ngamsWakeUpPlugIn(srvObj,
                      hostId):
    """
    Wake-Up Plug-In to wake up a suspended NGAS host.

    srvObj:         Reference to instance of the NG/AMS Server (ngamsServer).

    hostId:         Name of NGAS host to be woken up (string).

    Returns:        Void.
    """
    info(3,"Wake-Up Plug-In executing ...")
    hostDic = ngamsHighLevelLib.\
              getHostInfoFromHostIds(srvObj.getDb(), [hostId])
    if (not hostDic.has_key(hostId)):
        errMsg = "ngamsWakeUpPlugIn: Could not wake up host: " + hostId +\
                 " - host not defined in NGAS DB."
        raise Exception, errMsg

    networkDevs = srvObj.getCfg().getWakeUpPlugInPars()
    cmdFormat = "sudo /sbin/ether-wake -i %s -b " +\
                hostDic[hostId].getMacAddress()
    info(3,"Waking up suspended host: %s" % hostId)
    for dev in networkDevs.split(","):
        cmd = cmdFormat % dev
        info(3,"Broadcasting wake-up package - command: %s" % cmd)
        stat, out = commands.getstatusoutput(cmd)
```

```
        if (stat != 0):
            format = "ngamsWakeUpPlugIn: Problem waking up host: %s " +\
                     ". Error: %s."
            errMsg = format % (hostId, str(out).replace("\n", " "))
            raise Exception, errMsg
    info(3,"Leaving Wake-Up Plug-In")


# EOF
```

Figure 54: Example Wake-Up Plug-In (FILE: "ngams/ngamsPlugIns/ngamsWakeUpPlugIn.py").

1.67 EXPERT: The Filter Plug-in

The purpose of the Filter Plug-In is to classify data when data is being selected for delivery to a requestor for instance in connection with the Data Subscription Service (see section 1.29). The Filter Plug-In is a function which understands the data it is applied to, and which based on the contents of the data file, perhaps the filename and other pertinent information can determine if the file matches the requirements.

.0.25 EXPERT*:* Interface of a Filter Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Data Checksum Plug-In has an interface as shown in ***Figure 55***.

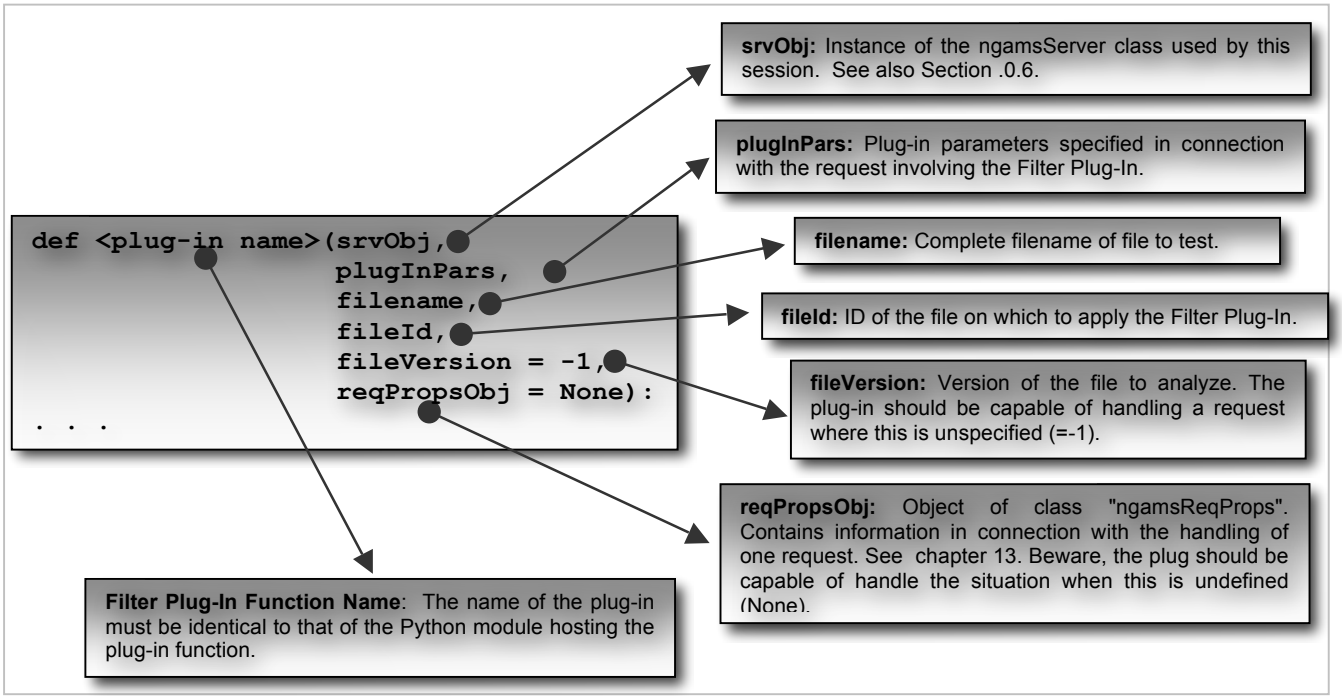

Figure 55: Function interface of a Filter Plug-In.

A Filter Plug-In must return either 1 or 0 depending on whether the filter conditions are met or not.

.0.26 EXPERT*:* Example Filter Plug-In

The following Python module, is the implementation of a simple Filter Plug-In, which filters the file specified according to a requested mime-type given in the Plug-In Parameter string ("mime_types", can contain a list of "|" separated mime-types).

```
#******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsMimeTypeFilterPI.py,v 1.8 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  21/11/2002  Created
```

```
#

"""
Contains a Filter Plug-In used to filter on the mime-type of the data.
"""

from ngams import *
import ngamsPlugInApi


def ngamsMimeTypeFilterPI(srvObj,
                          plugInPars,
                          filename,
                          fileId,
                          fileVersion = -1,
                          reqPropsObj = None):
    """
    Example Filter Plug-In used to filter on a given mime-type. In case the
    file referenced has the mime-type as specified in the plug-in parameters,
    the file being tested is selected.

    srvObj:        Reference to NG/AMS Server Object (ngamsServer).

    plugInPars:    Parameters to take into account for the plug-in
                   execution (string).

    fileId:        File ID for file to test (string).

    filename:      Filename of (complete) (string).

    fileVersion:   Version of file to test (integer).

    reqPropsObj:   NG/AMS request properties object (ngamsReqProps).

    Returns:       0 if the file does not match, 1 if it matches the
                   conditions (integer/0|1).
    """
    match = 0

    # Parse plug-in parameters.
    parDic = []
    pars = ""
    if ((plugInPars != "") and (plugInPars != None)):
        pars = plugInPars
    elif (reqPropsObj != None):
        if (reqPropsObj.hasHttpPar("plug_in_pars")):
            pars = reqPropsObj.getHttpPar("plug_in_pars")
    parDic = ngamsPlugInApi.parseRawPlugInPars(pars)
    if (not parDic.has_key("mime_types")):
        errMsg = "ngamsMimeTypeFilterPI: Missing Plug-In Parameter: " +\
                 "mime_types"
        raise Exception, errMsg

    # Perform the matching.
    refMimeTypes = parDic["mime_types"].split("|")
    actMimeType = ngamsPlugInApi.determineMimeType(srvObj.getCfg(), filename)
    for mt in refMimeTypes:
        if (actMimeType == mt.strip()): match = 1

    return match


# EOF
```

Figure 56: Example Filter Plug-In (FILE: "ngams/ngamsPlugIns/ngamsMimeTypeFilterPI.py").

1.68 EXPERT: The Disk Synchronization Plug-in

The purpose of the Disk Synchronization Plug-In is to ensure that the caches are purged to ensure that the file has arrived on the disk before updating the information about the file in the DB. If this is not ensured, there is a marginal case, whereby a file may be updated in the DB when it has actually not be stored completely on disk. If the node is shut down abruptly before the cache has been flushed, this means that a corrupt file has been archived.

	*If a Disk Synchronization Plug-In is not specified, NG/AMS will perform a normal/standard 'sync' on the UNIX/Linux shell.*

.0.27 EXPERT*:* Interface of a Disk Synchronization Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Disk Synchronization Plug-In has an interface as shown in ***Figure 51***.

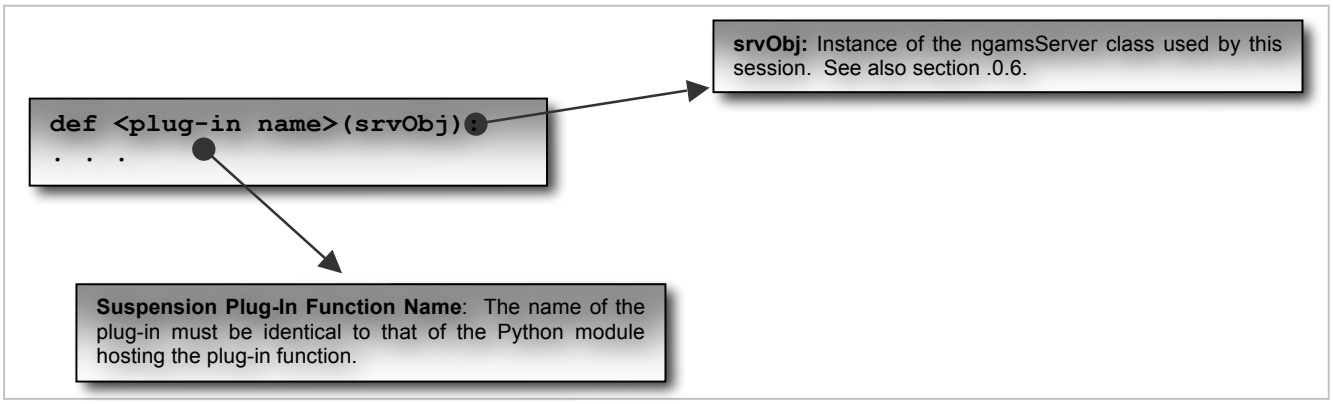

Figure 57: Function interface of a Disk Synchronization Plug-In.

The reason for supporting this type of plug-in is if a HW would be used for controlling disks, where it would be necessary to invoke a special, HW/firmware specific command to flush the caches.

.0.28 EXPERT*:* Example Disk Synchronization Plug-In

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngams3wareDiskSyncPlugIn.py,v 1.6 2004/08/04 16:45:30 ngasmgr Exp $"
#
# Who       When        What
# --------  ----------  ------------------------------------------------------
# jknudstr  10/05/2001  Created.
#

"""
Module that contains a Disk Sync Plug-In for the 3ware Controller.
"""

from   ngams import *
import ngamsPlugInApi


def ngams3wareDiskSyncPlugIn(srvObj):
    """
    Disk Sync Plug-In to flush the cache of the 3ware controller.

    srvObj:        Reference to instance of NG/AMS Server class (ngamsServer).

    Returns:       Void.
    """
    info(4,"Entering ngams3wareDiskSyncPlugIn() ...")

    # Sync filesystem to ensure file received on disk.
    info(3,"Performing OS sync command ...")
    commands.getstatusoutput("sync")
    info(3,"Performed OS sync command")
    logFlush()

    info(4,"Leaving ngams3wareDiskSyncPlugIn()")


if __name__ == '__main__':
    """
    Main function.
    """
    pass


# EOF
```

Figure 58: Example Suspension Plug-In (FILE: "/ngams/ngamsPlugIns/ngams3wareDiskSyncPlugIn.py").

1.69 EXPERT: The User Command Plug-in

TODO

1.70 EXPERT: The User Service Thread Plug-in

TODO

1.71 EXPERT: The Cache Control Plug-in

TODO

12. The NG/AMS Status XML Document

The NG/AMS Status Document is used in various contexts, either as the complete status or as partial status for a specific context. For instance, in the reply of most commands, a small status is given indicating if the command was executed successfully, and in case not, indicating the error that occurred.

1.72 EXPERT: NG/AMS Status DTD ("ngamsStatus.dtd")

The NG/AMS Status document is based on the NG/AMS base DTD described in section 1.46. The contents of this document can be displayed by following the link:

http://<Host>:<Port>/ngams.ngamsData.ngamsStatus_dtd.html

1.73 NGAS Disk Info Status - Example

Apart from keeping the information about an NGAS Disk in the NGAS DB (DB: "ngas_disks"), NG/AMS maintains a snap-host of this information in an XML document on each disk. This document is referred to as the "NgasDiskInfo" file. It is created when the disk is first registered, and subsequently updated each time the NG/AMS Server goes Online/Offline, and when the disk is completed.

The following is an example of an "NgasDiskInfo" file. Such XML status documents are stored on each NGAS disk.

```
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2003-01-02T08:40:23.350" HostId="acngast1" Message="Disk status file"
          Version="v2.0-Beta2/2002-12-04T09:22:53"/>
    <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32300" BytesStored="8709834319" Checksum=""
                Completed="0" CompletionDate="" DiskId="IC35L040AVER07-0-SXPTX093675"
                InstallationDate="2002-11-25T09:48:25.000" LogicalName="FITS-M-000001"
                Manufacturer="IBM" NumberOfFiles="163" TotalDiskWriteTime="905.324898006"
                Type="MAGNETIC DISK/ATA"/>
</NgamsStatus>
```

Figure 59: Example NGAS Disk Info file (FILE: "<mount root point>/<disk mount point>/NgasDiskInfo").

The NGAS Disk Info files are stored at the following location for each disk: "<mount root point>/<disk mount point>/NgasDiskInfo".

1.74 NGAS File Info Status - Example

The following is an example of a File Info Status document, which is generated e.g. when archiving a file or when issuing a "STATUS?file_id=<file ID>[&file_version=<file version>]" request:

```
<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus SYSTEM "http://acngast1.hq.eso.org:7777/RETRIEVE?internal=ngamsStatus.dtd">
<NgamsStatus>
  <Status Date="2003-01-02T13:48:49.758" HostId="acngast1"
          Message="Successfully handled command STATUS" State="ONLINE" Status="SUCCESS"
          SubState="IDLE" Version="v2.0-Beta2/2002-12-04T09:22:53"/>
  <DiskStatus Archive="ESO-ARCHIVE" AvailableMb="32374" BytesStored="8851908825" Checksum=""
              Completed="0" CompletionDate="" DiskId="IBM-DTLA-305040-YJ0YJ070913"
              HostId="acngast1" InstallationDate="2002-11-25T09:48:25.000" LastCheck=""
              LogicalName="FITS-R-000001" Manufacturer="IBM" MountPoint="/NGAS/data2" Mounted="1"
              NumberOfFiles="164" SlotId="2" TotalDiskWriteTime="350.22437346"
              Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1810827525" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="WFI.2001-09-15T22:49:07.652"
                FileName="saf/2001-09-15/1/WFI.2001-09-15T22:49:07.652.fits.Z"
                FileSize="142074506" FileStatus="00000000" FileVersion="1"
                Format="application/x-cfits" Ignore="0" IngestionDate="2003-01-02T13:48:10.000"
                Tag="" UncompressedFileSize="141546240"/>
  </DiskStatus>
</NgamsStatus>
```

Figure 60: Example File Info Status.

13. *EXPERT:* The NG/AMS Python Modules

In this chapter an overview of the NG/AMS Python modules, classes, functions and 'constants' is given. It is not the intention to provide the complete and detailed documentation for all this. This is contained as inline Python documentation in the Python code and can be browsed online. See section 1.76 for a description of how to do this. The purpose of this chapter is merely to give an overview of the NG/AMS Package.

For the basic usage of NG/AMS it is normally not necessary to have a deep knowledge about the internals of the SW. However, when developing the different types of plug-ins, which must be provided to adapt NG/AMS to various specific contexts, it is an advantage, and in some cases crucial, to have some insight in and overview of the SW and the classes and features available.

1.75 EXPERT: NG/AMS Module Structure

Although this manual is not meant as a maintenance manual for NG/AMS, the structure of the NG/AMS modules is briefly mentioned here. This information may be useful in case of troubleshooting or in general for obtaining a deeper insight into the system. The main NG/AMS project module contains the following files and modules (only items of interest in this context are listed):

Module/File	*Description*
__init__.py	The main Python module containing definitions of basic functions, and definition of various constants (variables).
LICENSE	File that contains the license and distribution conditions for the NG/AMS SW.
INSTALL	File that provides a small installation guide for the NG/AMS SW.
VERSION	Contains the version information for NG/AMS. This is the information that is printed on stdout when issuing the “-version” parameter to the NG/AMS Server or the command line utilities. This is also used in the XML status messages sent back as response to requests to the server.
ngamsCClient	The NG/AMS C based API. Also provides the C based command line utility.
ngamsData	Contains the definition of the various NG/AMS XML data formats. In addition various example files are provided.
ngamsLib	The base module provides various Python modules with fundamental functions, classes and methods used throughout the NG/AMS SW.
ngamsPClient	The NG/AMS Python based API. Also provides the Python based command line utility.
ngamsPlugIns	Contains various example plug-ins implemented for the usage of NG/AMS within ESO.
ngamsServer	Contains the source code used to build the NG/AMS Server.
ngamsSql	Contains the SQL scripts used to build the NGAS DB.
ngamsTest	Contains the NG/AMS Unit Test Suite.

Table 17: Files and modules in the NG/AMS project.

The "ngamsLib" module is the one, which a plug-in developer mostly will be concerned with, although some knowledge about the NG/AMS Server Class (and Python module) is also needed.

In the following some components of potential interest for plug-in development from the module “ngamsLib” are briefly described:

Python Module	*Class*	*Description*
ngamsConfig.py		Contains the code for the "ngamsConfig" class together with other classes used in connection with the NG/AMS Configuration. This is all used to handle the configuration programmatically.
	ngamsDppiDef	Contains the definition of one DPPI from the configuration.
	ngamsStorageSet	Class used to manage the information in connection with one Storage Set from the NG/AMS Configuration.
	ngamsStream	Class used to manage the information in connection with a Stream Definition from the NG/AMS Configuration.
	ngamsConfig	Class used to handle the information in the NG/AMS Configuration. It is possible to load and save the configuration file, as well as to setting and getting all properties of the configuration. It is also possible to generate an XML document of the configuration contained in the object.
ngamsDapiStatus.py		The module provides the class "ngamsDapiStatus", which is used to

		handle the status information from the execution of a Data Handling Plug-In (see chapter 1.61). An instance of this class is returned by a DAPI to the NG/AMS Server.
ngamsDb.py		The module provides the class "ngamsDb", which is used to access the NGAS DB. All DB access should be performed through this class. This therefore contains all the necessary SQL queries used by the NG/AMS SW. Many methods are provided to perform various, specific queries into the NGAS DB. A 'native SQL query' can be performed using the method "ngamsDb.query()".
ngamsDiskInfo.py		Provides the class "ngamsDiskInfo", which is used to handle all the information in connection with an NGAS disk. The object can also contain information about the files on the disk. This is stored internally as "ngamsFileInfo" objects. It is possible to generate an NG/AMS XML Status document from the contents of the object.
ngamsDiskUtils.py		Functions used to carry out the handling/management of the disks installed. Among this are function to extract the information about the disk configuration, and a function to check the accessibility of the disks installed.
ngamsDppiStatus.py		The module provides the class "ngamsDppiStatus", which is used to handle the status information from the execution of a Data Processing Plug-In (see chapter 1.63). An instance of this class is returned by a DPPI to the NG/AMS Server.
	ngamsDppiResult	Class that contains a sub-result from a DPPI execution.
	ngamsDppiStatus	Class that contains the resulting data from a DPPI execution.
ngamsFileInfo.py		The module provides the class "ngamsFileInfo", which is used to handle all the information in connection with a file, which has been archived in an NGAS Host. It is possible to generate an XML document from the contents of the object.
ngamsFileList		Used to manage list of file information objects (ngamsFileInfo), e.g. to dump the information into XML documents.
ngamsLib.py		Contains various basic convenience functions used throughout the NG/AMS SW.
ngamsNotification.py		Contains functions to manage the Email Notification feature. This also includes tools for handling the Email Notification Retention.
ngamsPhysDiskInfo.py		Provides the class "ngamsPhysDiskInfo", which is used to manage the 'physical information' about a disk extracted by the System Online Plug-In (see chapter 1.58).
ngamsPlugInApi.py		Modules that provides various utility functions to be used for implementing plug-ins. It is recommended only to use the functions contained in this module for implementing the plug-in, apart from varios classes like ngamServer, ngamsDb and ngamsConfig.
ngamsReqProps.py		Module that provides the object "ngamsReqProps", which is used to keep a record of actions carried out during the handling of a request.
ngamsStatus.py		Provides the class "ngamsStatus", which is used to handle the information in connection with a status generated for NG/AMS.
ngamsSubscriber.py		Used to handle information about one Subscriber.

Table 18: Python modules in the "ngamsLib" sub-module.

1.76 EXPERT: Online Browsing of NG/AMS Inline Python Documentation

It is possible to browse online the Python documentation contained in the NG/AMS Python source code files. This provides an accurate and comprehensive description of all classes, methods and functions. The following notation has been used to document the interfaces of methods and functions, e.g.:

```
def notify(ngamsCfgObj,
           type,
           subject,
           msg):
    """
    Send a notification e-mail to a subscriber about an event happening.

    ngamsCfgObj:   Reference to object containing NG/AMS Configuration file (ngamsConfig).

    type:          Type of Notification (See NGAMS_NOTIF_* in ngams).
```

```
subject:        Subject of message (string).

msg:            Message to send (string).

Returns:        Void.
"""
<code>
```

Figure 61: Example of NG/AMS inline documentation.

First in the description of a method/function, a small description of the task performed by the method is provided. After that the input parameters are listed. After the description of each parameter the type of the parameter is indicated in paranthesis. The return value is also given in connection with the "Returns:" tag.

The documentation can be browsed in an easy manner by using the documentation generator provided together with the Python package ("pydoc"). This can also be used as an HTTP server, e.g.:

```
ngasdev2 jknudstr:~/ 65 > pydoc -p 7575 &
[2] 15578
ngaadev2 jknudstr:~/ 66 > pydoc server ready at http://localhost:7575/
```

Figure 62: Starting the pydoc utility as an HTTP server.

Afterwards the NG/AMS documentation can be accessed online via the URL (e.g.):

http://<Host>:<Port>/ngams.html

The pydoc utility provides a convenient way of browsing the documentation, and generates the documentation online. It locates the NG/AMS module if installed within the search paths compiled into the Python interpreter. Alternatively it should be located in a path defined in the "PYTHON_PATH" environment variable.

14. *EXPERT:* Installation

The installation of the NGAS Package goes somehow beyond the scope of this manual since this normally deals only with NG/AMS and not the complete system as such. Nevertheless, in this chapter it is described in short how to install the NGAS Package either automatically or manually.

1.77 EXPERT: Automatic Installation of NGAS

From NGAS V2.3 tools have been provided to install an NGAS host from scratch, in an automatic way. The automatic installation has been split into two parts:

1. **Installation of OS:** This part of the installation is carried out with a so-called kickstart script. It installs a basic RedHat 9.0 platform with selected packages needed by the NGAS Packages. After executing this script, a generic NGAS OS platform has been obtained. No specific NGAS features are supported. The kick start script relies on downloading packages from a server within the ESO network and network access must therefore be available. There is such a kick start file per node, which is installed from scratch.
2. **Installation of NGAS Package:** The second part of the installation consists in installing the NGAS layer on top of the OS platform. This is done by means of the generic installation script (FILE: “ngasInstall/src/ngasInstallSystem.py”) provided by the NGAS Package:

   http://jewel1.hq.eso.org:7575/ngasInstall.src.ngasInstallSystem.html

   This NGAS Installation Script must be executed as user root. It will prompt for various information during the installation procedure. It is also possible to run the tool as a one-line command, whereby this information is provided on the shell when issuing the command.

Also the NGAS Installation Script relies on downloading packages from a WEB site within the ESO network

If it is desirable to obtain information about this topic or get access to the kickstart scripts, which install Linux, please contact the NGAS Team: ngast@eso.org.

1.78 EXPERT: Manual Installation of NGAS

The manual installation of an NG/AMS Server *can* be a relatively straightforward and simple procedure if it is not necessary to create an adapted installation for a specific context. Under normal circumstances, the most complex action in this connection might be to configure the server properly. The steps to carry out to obtain a running NG/AMS installation are as follows:

Step/Action	*Description*
Verify RDBMS Installation	Verify that the RDBMS used in connection with this NGAs Unit, is properly installed and the RDBMS server running and accessible (if relevant). Verify also that the client libraries used to access the RDBMS, are available.
Install Python, Check Existing Python Installation	The present version of Python required for NG/AMS is 2.1 (2.1.1). Check that the proper version is installed. If the wrong version is installed, or if Python is not installed at all, it should be downloaded from **http://www.python.org** and installed according to the instructions. Check in particular that the Sybase Python module is available (if Sybase is used).
Get the NG/AMS Python SW	Get the sources of the NG/AMS SW. This can be requested by contacting: **ngast@eso.org**.
Install NG/AMS SW + Configure the Environment	Install the sources simply by copying the NG/AMS root module directory "ngams" to a path contained within the "PYTHON_PATH" list of paths, or add the new location of "ngams" in the "PYTHON_PATH" variable. The NG/AMS C-API should also be compiled and installed. This is done by entering in the directory "ngams/ngamsCClient" and typing "make clean all". The binary "ngamsCClient" should be installed in a 'bin' directory for global access. The "ngams.h" and "libngams.a" files should be copied into an area which is globally accessible (if needed for application development). It could also be chosen to make the NG/AMS Server source file ("ngams/ngamsServer/ngamsServer.py") executable and globally accessible. The same goes for the NG/AMS Python API ("ngams/ngamsPClient/ngamsPClient.py").

Prepare Sybase DB	Prepare the NGAS DB in the Sybase DB server. A user to be used by the NG/AMS when connecting to the DB should be created, e.g.: "ngas". The NGAS tables must also be created. This should be done using the SQL script contained in "ngams/ngamsSql". The script is called: "ngamsCreateTables.sql. The script can be executed using "isql".
Prepare NG/AMS Configuration	Use possibly as a template configuration the configuration example file provided within the NG/AMS SW package ("ngams/ngamsData/ngamsServer.xml"). Go carefully through the list of parameters and configure these according to the description provided in chapter 6).
Prepare Plug-Ins	Prepare the necessary plug-ins needed for operating NG/AMS. The plug-ins to consider first are the System Online and Offline Plug-Ins; see the chapters 1.58 and 1.59. In addition the DAPI for each type of data to be handled (archived); see chapter 1.61. If it is desirable to calculate a checksum for the data files being archived, a Data Checksum Plug-In must be provided; see chapter 1.64. If data should be processed, a DPPI should be provided for each type of processing offered by the system; see chapter 1.63. If labels for the disk cases should be generated, a Label Printer Plug-In must be provided as well; see chapter 1.60. Example implementations of all of these types of plug-in are provided within the NG/AMS package ("ngams/ngamsPlugIns"). Note that all plug-ins provided should be made available in a path pointed to by the "PYTHON_PATH" variable or in one of the search paths compiled into the Python interpreter.
Launch Server in Simulation Mode	The first time when the NG/AMS Server is started after doing all the necessary configuring, it may be convenient to start it manually in an xterm in the Verbose Mode (Verbose Level 3 or 4); see also section 1.40. This could be done in Simulation Mode to first get the basic things straightened out. If the server encounters problems, it will bail out and report these on “stdout”. The switching on/off of the Simulation Mode/Normal Mode must be done in the NG/AMS Configuration. It could be tried to issue some commands like ARCHIVE and RETRIEVE to verify the proper functioning.
Launch server in real mode	When the server is running properly in Simulation Mode, it could be tried to switch to the Normal Mode (in the configuration), and try the same as described in the previous step.
Decide how to Start the Server	When the server can be executed and is operating correctly, it should be decided how it should be started. Under normal circumstances it should be started when the host on which it is running is booting, and run as a daemon. I.e., the start-up scripts on the host should be configured accordingly.
Handling of Local Log Files	If a Local Log File is generated, it should be considered that this will continuously grow in size. The speed with which it will be growing, depends on the Log Level selected. If it is desirable to keep the log files, a DAPI to handle this could be provided for NG/AMS and a cron job launched periodically to archive the log file into NG/AMS and subsequently to delete it. If it is not desirable to preserve the information, the file could be deleted periodically. This however, is up to the people responsible for the individual installation to decide how to handle this.
Configuring of Security Mechanisms	Since no security mechanisms are provided at the level of NG/AMS to prevent 'intruders' to connect to the server, such mechanisms should be put in place at the level of the operating system or network. It is up to the people responsible for the security in connection with IT services to decide how to implement this.
Setting Up of Multi-Site DB Environment	If an organization is running a distributed NGAS system, whereby data e.g. are produced on several remote sites, and are made available in an Archive Facility, considerations should be done as how to set up the DB infrastructure. It might be most logical to have the central/reference DB in connection with the Archive Facility, and to set up replication from the various remote sites to the Archive Facility DB. See also section 1.52 for more information about this issue.

Table 19: Steps needed to install NG/AMS.

As can bee seen above, the installation in the worst-case may be a quite complex procedure. It is therefore not feasible to provide a complete and detailed information in the NG/AMS User's Manual about this. In case of problems or questions it is suggested to contact: **ngast@eso.org** for advice and help on how to approach this matter.

15. NG/AMS Log and Error Messages Definition

Many important log messages (information) and error messages are defined in a formal way in an XML document (FILE: “ngams/ngamsData/ngamsLogDef.xml”). Defining these log messages in this way makes it possible for a client application to better parse/analyze a reply sent back from NG/AMS, since the log definition is based on tags and error code, which will appear in the message.

The logs defined in the NG/AMS Log Definition Document, must be used together with the global NG/AMS function “genLog()”. An example is shown here:

```
errMsg = genLog("NGAMS_ER_DAPI", ["Unknown compression method specified: " + comprMethod])
raise Exception, errMsg
```

Figure 63: Simple example of generating a log using the NG/AMS Log Definition.

Another, example is the following where several parameter are fed into the “genLog()” function to build up the log message and where also the generated error message is logged on the specified log targets:

```
action = “REMDISK”
errMsg = genLog("NGAMS_ER_IMPROPER_STATE",[“REMDISK”, self.getState(), self.getSubState(),
            str(allowedStates), str(allowedSubStates)])
error(errMsg)
raise Exception, errMsg
```

Figure 64: Example of generating a log using the NG/AMS Log Definition.

The complete log definition can be obtained by following the link:

http://<Host>:<Port>/NGAS/ngamsLogDef.html

The format of the log mnemonics is: <Log Type>_<Action>. The following prefixes are used in the NG/AMS Log Definition:

Log Type/Prefix	*Description*
*NGAMS_AL_*	Corresponds to UNIX Alert type of log (LOG_ALERT).
*NGAMS_ER_*	Corresponds to UNIX type of log (LOG_ERR).
*NGAMS_INFO_*	Corresponds to UNIX type of log (LOG_INFO).
*NGAMS_NOTICE_*	Corresponds to UNIX type of log (LOG_NOTICE).
*NGAMS_WA_*	Corresponds to UNIX type of log (LOG_WARNING).

Table 20: Mapping of UNIX log types and log prefixes in the NG/AMS Log Definition.

Note, the log types “LOG_CRIT”, “LOG_DEBUG”, “LOG_EMERG” and “LOG_EMERG” are not used for the moment within the context of the NG/AMS logging system for predefined log formats.

16. NG/AMS Commands

This chapter contains a detailed reference to the commands supported by the NG/AMS Server. All the commands are listed and explained, and the command parameters in connection with these are described.

Using the NG/AMS APIs (chapters 9 and 10) or the NG/AMS Python or C based command line utilities (section 1.41), the user is assisted in applying the proper parameters. It is recommended to use these when communicating with the NG/AMS Server.

1.79 ARCHIVE Command - Archive Data Files

The ARCHIVE Command is used to archive data files. The ARCHIVE Command accepts the following parameters:

Parameter	*Mandatory*	*Description*
filename=<file URI>	Yes	The parameter is used to specify the location of the file. In case of an Archive Push Request, NG/AMS may use the given URI, to determine the mime-type of the file. In addition, the temporary filename in the Staging Area is based on the filename (without the path) given in the URI. For an Archive Push Request, the URI is the location (URL) where the file can be picked up by the NG/AMS Server. The location must then be accessible from the NG/AMS Server either via HTTP (http://…), FTP (ftp://…) or directly as file (file://…). Also in this case the mime-type pf the data may be determined from the path if not specified directly by means of the "mime_type" parameter.
mime_type=<mime-type>	No	If the File URI of an Archive Request does not reveal the mime-type of the file to be archived, the mime-type should be specified in the Archive Request. This makes the handling of the request more efficient. An example using this parameter is given in Example 3 below.
no_versioning	No	Used to switch the automatic versioning on/off. If File Versioning is on, a file archived with a File ID already registered in the NGAS DB, will get a new version number (previous number + 1).
wait=0|1	No	With this parameter it is possible to specify if the NG/AMS Server should send back an immediate reply (wait=0) when handling an Archive Request, or if a reply should be sent after the request has been handled (wait=1). In the former case, the client will not know if the Archive Request was handled successfully. The default behavior of the server is to send the *reply* after the Archive Request has been handled.

Table 21: Parameters for the ARCHIVE Command.

Example 1: Archiving using Archive Push Technique:
An example of an Archive Push Request can be found in section 1.48/***Figure 18***.

Example 2: Archiving using Archive Pull Technique:
The URL for the NG/AMS Server could be something like this:

> http://hostx:7878/ARCHIVE?filename=ftp://hosty/data/2002-02-11/Fits1.fits[10]

In this case the NG/AMS Server will pick up the file from the location given. I.e., the client need not to issue the data in the HTTP request. In the example shown, the NG/AMS Server will generate a reply after having handled the Archive Request.

Example 3: NGAS Node to NGAS Node Archiving:
As a small 'curiosity', this example shows an Archive Pull Request, whereby the file URI specified is referring to a file located on another NGAS Node:

> http://ngas1:7878/ARCHIVE?mime_type=application/fits&filename=http://ngas2:7878/RETRIEVE?file_id=XYZ-2002-02-01T02:23:41.342

In this example, the NG/AMS Server handling the Archive Request will pick up the file transparently from the remote NGAS Host using the file URI which in this case is a Retrieve Request, and will archive it.

[10] The HTTP query string must be encoded according to the specification of the HTTP protocol. Here they are shown un-encoded.

1.80 CACHEDEL Command – Remove a File from an NGAS Cache Archive

The CACHEDEL Command is used to remove a file explicitly, which has been archived into on an NGAS Node, operated as a cache archive. Submitting the CACHEDEL Command to remove a file, overrules the conditions which would normally be applied, when deciding whether to remove the file or not.

The parameters for the CACHEDEL Command are:

Parameter	*Mandatory*	*Description*
disk_id=<disk ID>	Yes	The Disk ID of the files to consider.
file_id=<file ID>	Yes	The ID of the files to consider.
file_version=<file version>	Yes	The File Version of the files to consider.

Table 22: Parameters for the CACHEDEL Command.

The CACHEDEL Command supports proxy mode, such that a CACHEDEL Command, dedicated to a certain file, not hosted on the contacted node, may be forwarded transparently. It is only the "ngamsCacheServer" version of the NG/AMS Server, which supports the CACHEDEL Command however.

1.81 CHECKFILE Command – Check File Consistency

The CHECKFILE Command is used to check the consistency of a specific file. This corresponds to the checks carried out by the NG/AMS Data Consistency Check, in this case though only on one selected file.

The parameters for the CHECKFILE Command are:

Parameter	*Mandatory*	*Description*
disk_id=<disk ID>	Yes	The Disk ID of the files to consider.
file_id=<file ID>	Yes	The ID of the files to consider.
file_version=<file version>	Yes	The File Version of the files to consider.
host_id=<host ID>	No	ID of host where the file to check is stored.

Table 23: Parameters for the CHECKFILE Command.

1.82 CLONE Command – Create File Copies

The CLONE Command is used to create copies of single files or sets of files. In order for the CLONE Command to be accepted by an NG/AMS Server, the system must be configured to accept Archive Requests. Also, enough free disk space must be available in the NGAS Host handling the request. NG/AMS will calculate if there is enough space to execute the request. If the files to be cloned are located on other NGAS Hosts, these will be requested automatically during the cloning (if possible). If such hosts are suspended, they will be woken up automatically. If a node hosting a file to be cloned is within a cluster different than the one to which the node executing the Clone Request belongs, the Master Unit for the given node will be contacted.

The files to be cloned are selected, based on the parameters File ID, Disk ID and File Version. The interpretations of the various combinations of these parameters are explained in ***Table 24***.

Disk ID	*File ID*	*File Version*	*Interpretation*
-	+	-	Clone one file with the given ID. Latest version of the file is taken.
+	+	-	Clone one file stored on the given disk. Latest version on that disk is taken.
-	+	+	Clone all files found with the given File Version. Storage location (Disk ID) is not taken into account.
+	+	+	Clone one file on the given disk with the given File Version.
+	-	-	Clone all files from the disk with the given ID.
+	-	+	Clone all files with the given File Version from the disk with the ID given.
-	-	+	Illegal. Not accepted to clone arbitrarily files given by only the File Version.
-	-	-	Illegal. No files specified.

Table 24: Rules applied when selecting files for cloning.

The only way in the present version to abort a Clone Request in progress, is to send an “OFFLINE –force” to the server (a “CLONE –abort” might be provided at a later stage). Note, in this version of NG/AMS it is not possible to specify a target disk for the cloning. This will be provided at a later stage.

The parameters for the CLONE Command are:

Parameter	*Mandatory*	*Description*
disk_id=<disk ID>	See ***Table 24***	The Disk ID of the files to consider.
file_id=<file ID>	See ***Table 24***	The ID of the files to consider.
file_version=<file version>	See Table 24	The File Version of the files to consider.
notif_email=<email list>	No	List of comma separated email addresses to where the Clone Status Report can be send.
wait=0|1	No	Indicate to NG/AMS whether or not it should return a response when the complete Clone Request has been executed or only, when the initial check if the command can be executed has been carried out.

Table 25: Parameters for the CLONE Command.

As a result of the CLONE Command, a File Cloning Status Report can be send back on request, indicating which files were cloned and the target names of these.

1.83 CONFIG Command – Change Configuration Online

The CONFIG Command is used to change the configuration of the NG/AMS Server while this is running. This can be used e.g. to increase the log level to obtain more information, without restarting the server.

The parameters for the CONFIG Command are:

Parameter	*Mandatory*	*Description*
log_sys_log=<level>	No	Change the Sys Log Level.
log_sys_log_prefix=<prefix>	No	Change the prefix for the Sys Logs.
log_local_log_file=<file>	No	Change the name of the Local Log File.
log_local_log_level=<level>	No	Change the Log Level for logging into the Local Log File.
log_verbose_level=<level>	No	Change the Verbose Level.
log_buffer_size=<size>	No	Change the Log Buffer Size, i.e., the internal log cache.

Table 26: Parameters for the CONFIG Command.

For the moment only a limited set of parameters are supported. This may be extended in the future.

1.84 DISCARD Command – Force Suppression of Files

Force suppression of files in the system. This may either be files archived and thus registered in the NGAS DB or stray files' on the data disks, referred to by their complete filename.

	*Extreme care should be taken when executing the DISCARD Command: Unlike the REMFILE and REMDISK Command, which checks if there are at least 3 copies of each file to be deleted, DISCARD simply deletes them. If executed with the "execute" parameter though, it will not delete the file, but will first issue a warning.*

The DISCARD Command should normally only be used when it is absolutely necessary to remove a data item, which cannot be removed by means of the REMFILE/REMDISK Commands.

The parameters of the DISCARD Command are:

Parameter	*Mandatory*	*Description*
disk_id=<ID>	No/Yes when file_id given	The Disk ID of the disk hosting the file to be deleted.
file_id=<ID>	No	The File ID of the file selected for deletion.
file_version=<version>	No/Yes when file_id given	Version of the file to be deleted.
host_id=<host ID>	No	
Path=<path>	No	
Execute=0|1	No	

Table 27: Parameters for the DISCARD Command.

In order for the DISCARD Command to be accepted the system must be configured to allow remove requests (CFG: “NgamsCfg.Server:AllowRemoveReq”).

1.85 EXIT Command - Terminate Server

The EXIT Command is used to make the NG/AMS Server exit. The EXIT Command does not accept any parameters.

1.86 HELP Command – Acquire Online Help

Acquire help about the command interface. This command is not yet implemented.

1.87 INIT Command - Re-Initialize the System

The INIT Command is used make the NG/AMS Server re-initialize. This means that it will first go Offline, load the configuration and subsequently go Online. The INIT Command does not accept any parameters.

1.88 LABEL Command - Generating Disk Labels

The LABEL Command is used to print out labels to be put on the disk cases. The label is the Logical Name of a disk. The LABEL Command accepts the following parameters:

Parameter	*Mandatory*	*Description*
slot_id=<slot ID>	Yes	The ID of the slot in which the disk is installed.
host_id=<host ID>	No	The host in which the disk is installed. If this is not specified, the local host is assumed.

Table 28: Parameters for the LABEL Command.

An example of a label generated by NG/AMS (by means of the Label Plug-In), can be found in section 1.24.

1.89 OFFLINE Command - Bring System to Offline State

The OFFLINE Command is used to make the NG/AMS Server go Offline. The OFFLINE Command accepts the following parameter:

Parameter	*Mandatory*	*Description*
force	No	Force the system to Offline State even though an action is in progress like file cloning.

Table 29: Parameters for the OFFLINE Command.

Usage of the “force” option should be done with great care, as operations may be interrupted before termination, leaving the system in an ‘undefined’ condition.

1.90 ONLINE Command - Bring System to Online State

The ONLINE Command is used to make the NG/AMS Server go Online. The ONLINE Command does not accept any parameters.

1.91 REARCHIVE Command - Archive Previously Archived File

The purpose of the REARCHIVE Command is to register a file in the NGAS DB, whereby the NGAS meta-information about the file, has already been generated previously, when the file was archived with the normal ARCHIVE Command. This means that the process of extracting the meta-information and other processing can be skipped whilst *re-archiving* the file, which makes the processing more efficient.

The meta-information about the file is contained in the special HTTP header named “NGAS-File-Info”. It is stored as a base64 encoded NGAS XML representation information for the file (NGAS File Info). This encoding can be accomplished by means of the Python module “base64” (function “base64.b64encode()”).

The command does not require any parameters.

The data to be re-archived, should be contained in the body of the HTTP request as is the case for an Archive Push Request or a valid URL be provided from where the contents of the file can be retrieved provided, as is the case for the Archive Pull Request; see description of normal ARCHIVE Command.

A checksum check of the final destination file is carried out, against the checksum contained in the NGAS File Info, contained in the Re-archive Request. This should ensure with a fairly high probability, that the file re-archived is consistent with the previously archived version.

1.92 REGISTER Command - Register Existing Files on a Disk

The REGISTER Command is used to register files already stored on an (NGAS) disk. It is possible to register single files, or entire sets of files by specifying a starting path from which NG/AMS will look for files. Only files that are known to NG/AMS (with a mime-type defined in the configuration), will be taking into account. It is also possible to explicitly specify a comma separated list of mime-types that will be registered. Files with other mime-types than specified in this list will be ignored.

Parameter	*Mandatory*	*Description*
mime_type=<mime-type>	No	Comma separated list of mime-types to take into account. A single mime-type can also be specified.
path=<file path>	Yes	The starting path under which NG/AMS will look for candidate files to register. It is also possible to specify a complete path, whereby only a single file will be registered.
wait=0|1	No	Wait issuing response till complete command execution has finished. Otherwise, issue immediately a response as soon as the command was accepted.
notif_email=<email list>	No	Send an Email Notification Email as a report with the files registered.

Table 30: Parameters for the REGISTER Command.

As a response to the REGISTER Command, a Registration Report may be generated, indicating which files where registered.

1.93 REMDISK Command – Remove Information about Disks

The REMDISK Command is used to remove information about entire disks from NGAS. *Great caution should therefore be applied when using this command!* Both the information about the Storage Media and the files stored on, will be removed. NG/AMS will not accept to remove a file from the system unless there are at least three (3) independent copies of the file. Three independent copies refers to three copies of the file stored on three independent Storage Media. In order for the REMDISK Command to be accepted the system must be configured to allow remove requests (CFG: “NgamsCfg.Server:AllowRemoveReq”). If the command is executed without the “execute” parameter, the information about the disk is not deleted, but a report is generated indicating what will be deleted if the execution is requested (execute=1).

The REMDISK Command takes the following input parameters:

Parameter	*Mandatory*	*Description*
disk_id=<disk ID>	Yes	Disk ID for the disk to remove.
execute=0|1	No	If execute is not specified or specified as 0, no information will be deleted, but a report will be send back to indicate what will be deleted if the command is executed. If execute is specified as 1, the information in the DB and on the disk itself is deleted.
notif_email=<email list>	No	Comma separated list of email recipient, who will be informed about the execution of the command, i.e., which files are successfully/unsuccessfully registered.

Table 31: Parameters for the REMDISK Command.

As a result of the REMDISK Command, a report is send indicating which disk that was removed from the system.

1.94 REMFILE Command – Remove Files from the System

The REMFILE Command is used to remove information about files and the files themselves from NGAS. *Great caution should therefore be applied when using this command!* NG/AMS will not accept to remove a file from the system unless there are at least three (3) independent copies of the file. Three independent copies refers to three copies of the file stored on three independent storage media. In order for the REMFILE Command to be accepted, the system must be configured to allow remove requests (CFG: “NgamsCfg.Server:AllowRemoveReq”). If the command is executed without the “execute” parameter, the

information about the file(s) is not deleted, but a report is generated indicating what will be deleted if the execution is requested (execute=1).

The selection of the files to be scheduled for deletion is done based on the parameters Disk ID, File ID and File Version. The rules for this are shown in ***Table 32***.

Disk ID	File ID	File Version	Interpretation
+	-	-	Illegal.
+	+	-	All files with the given File ID on the disk with the given ID will be selected. No specific File Version will be taken into account.
+	-	+	No files are selected.
+	+	+	The referenced file with the given File ID and File Version on the given ID is selected (if this exists).
-	+	-	All files matching the given File ID pattern on the contacted NGAS Host are selected.
-	+	+	All files with the given File ID pattern and the given File Version are selected without taking the Disk ID into account.
-	-	+	No files are selected.
-	-	-	No files are selected.

Table 32: Selection rules applied for the REMFILE Command.

The REMFILE Command takes the following input parameters:

Parameter	Mandatory	Description
disk_id=<disk ID>	See ***Table 32***	Disk ID for the disk to remove.
execute=0|1	No	If execute is not specified or specified as 0, no information will be deleted, but a report will be send back to indicate what will be deleted if the command is executed. If execute is specified as 1, the information in the DB and on the disk itself is deleted.
file_id=<file ID>	See ***Table 32***	ID of files to take into account.
file_version=<file version>	See ***Table 32***	Version of files to take into account.
notif_email=<email list>	No	Comma separated list of email recipient, who will be informed about the execution of the command, i.e., which files are successfully/unsuccessfully registered.

Table 33: Parameters for the REMFILE Command.

As a result of the REMFILE Command, a report is send back, indicating which disk that were moved from the system, or alternatively, if the “execute” is 0 or unspecified, a list of files that will be deleted if the command is executed is returned.

1.95 RETRIEVE Command - Retrieve & Process Files

The RETRIEVE Command is used to retrieve archived data files from an NGAS Node. The RETRIEVE Command accepts the following parameters:

Parameter	Mandatory	Description
file_id=<file ID>	No	ID of file to retrieve.
file_version=<file version>	No	Version of the file to retrieve.
internal=<filename>[&host_id=<ID>]	No	Retrieve the contents of an internal file. Could e.g. be the a source file or the log definition file. This is mostly intended for maintenance/trouble-shooting purposes. If a Host ID is given, the file will be picked up from the referenced host.
ng_log[&host_id=<ID>]	No	Retrieve the contents of the NG/AMS Local Log File (see 1.17). If a Host ID is given, the log file is picked up from the referenced host.
cfg[&host_id=<ID>]	No	Retrieve the contents of the NG/AMS Configuration used. If a Host ID is given, the configuration is picked up from the referenced host.
processing_pars=<DPPI>	No	With this parameter it is possible to specify a DPPI, which is invoked to process the data before sending it back. NG/AMS will send back the result of the processing, and not the original file.

Table 34: Parameters for the RETRIEVE Command.

It is possible to receive an HTTP redirection response as response to the Retrieve Request. In this case the client must re-send the Retrieve Request to the alternative URL given in the redirection response. See also section 1.50.

If the command “RETRIEVE?internal=<folder>” is given, an XML document will be returned (FileList), with the contents of the given folder.

If a file requested is stored on a sub-node within a cluster, the file may be retrieved via a Master Unit of the cluster. If the sub-node hosting the file is suspended, it will be woken up by the master.

1.96 STATUS Command - Query System Status & Other Information

The STATUS Command is used to query various status information from the NG/AMS Server. The STATUS Command accepts the following parameters:

Parameter	*Mandatory*	*Description*
<no parameters> *[host_id=<ID>]*	No	In this case a reply is returned which contains an NG/AMS Status document. An example of such a status document can be found in section 1.22. If a host ID is given, the status of the referenced host will be retrieved and returned. If the specified host is suspended, it will be woken up.
disk_id=<disk ID>	No	Query information about a disk referred to by its Disk ID. The reply is an NG/AMS Disk Status XML document. An example of this can be found in section 1.73.
file_id=<file ID>	No	Query information about a file with a given File ID. The reply is an NG/AMS XML Status document as shown in section 1.74.
file_version=<file version>	No	Used to specify specific version of file to query information for.
configuration_file	No	Query the configuration used by an NG/AMS server. The result is a complete NG/AMS Configuration XML document as shown in section 1.47.
file_access=<file ID>	No	This parameter is used to make an NG/AMS Server probe if it can physically access a file. The body of this request must contain an NG/AMS File Status XML document with the detailed information about the file. It is therefore necessary to issue also the "content-length" and "content-type" HTTP headers followed by the file status in the request. This command is thus similar to an Archive Push request, refer to section 1.48 for further information about this issue.
flush_log	No	Used to make the NG/AMS Server flush the logs it may have cached internally, into the Local Log File if such is specified.
request_id=<ID>	No	Retrieve the status of a given request.

Table 35: Parameters for the STATUS Command.

It is only possible to specify one of the parameters at a time.

1.97 SUBSCRIBE Command – Subscribe to Data from NGAS Host

The SUBCRIBE Command is used by a Data Subscriber to subscribe to a certain kind of data becoming available (and which is available at the time of the subscription). The issue of Data Subscription is described in detail in section 1.29. The parameters of the SUBSCRIBE Command are listed in ***Table 36***.

Parameter	*Mandatory*	*Description*
filter_plug_in=<plug-in>	No	Name of a Filter Plug-In (see chapter 1.67) to invoke on the data to determine whether to deliver this to the client or not.
plug_in_pars=<pars>	No	A set of parameters to transfer to the Filter Plug-In when it is invoked.
priority=<prio>	No	Priority for delivering data to this Data Subscriber. The lower the number, the higher the priority. Clients with a higher priority, get more CPU time in connection with the data delivery.
start_date=<ISO8601>	No	Date from which the data to deliver is taken into account. If not specified the time when the SUBSCRIBE Command was received is taken as start date.
url=<delivery URL>	Yes	The URL to which the data will be delivered. On the client side a corresponding HTTP server must be ready to receive requests (data) via the given URL.

Table 36: Parameters for the SUBSCRIBE Command.

Note, a Data Subscriber that has subscribed remains subscribed also if the HTTP server, which handles the receiving of data on the client side terminates. NG/AMS will back-log data for that client, and when it re-subscribes, the back-logged data will be delivered. If it not desirable, the client should un-subscribe it-self by submitting an UNSUBSCRIBE Command.

1.98 UNSUBSCRIBE Command – Unsubscribe a Previous Data Subscription

Used by Data Subscribers to un-subscribe a previously established subscription for data. If NG/AMS holds back-logged buffered subscription data for the client from a previous subscription, the Subscription Back-Log will be reset. The parameter of the UNSUBSCRIBE Command is listed in Table 37

Parameter	*Mandatory*	*Description*
url=<delivery URL>	Yes	The URL, which was submitted when the client subscribed itself. The Delivery URL is used by NG/AMS to identify each Data Subscriber.

Table 37: Parameters for the UNSUBSCRIBE Command

If the client was not subscribed, the UNSUBSCRIBE Command has no effect.

17. Index

M

N

O

P

Q

R

S

T

U

V

W

X